Escola Politècnica Superior de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FIN DE CARRERA

TÍTULO DEL TFC: LEGO Mindstorms NXT como herramienta de aprendizaje en el ámbito de la ingeniería

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad Sistemas de Telecomunicación

AUTORA: Sandra Ramil Gomez

DIRECTOR: Marcos Quílez Figuerola

FECHA: 9 de octubre de 2009

Título: LEGO Mindstorms NXT como herramienta de aprendizaje en el ámbito de la ingeniería

Autora: Sandra Ramil Gomez

Director: Marcos Quílez Figuerola

Fecha: 9 de octubre de 2009

Resumen

Este proyecto tiene como objetivo principal el estudio del kit LEGO Mindstorms NXT y valorar su posible uso como herramienta de aprendizaje en estudios de ingeniería. Para ello, el proyecto se ha dividido en dos partes. En la primera se estudia la herramienta en sí. La segunda presenta diversas actividades, de dificultad creciente, que permiten introducir el LEGO NXT en estudios del ámbito de la ingeniería que utilicen la metodología de aprendizaje basado en proyectos (PBL).

En la primera parte, se ha realizado un estudio de cada una de las piezas del pack de LEGO: el brick, los motores y los sensores. Además, se ha realizado el estudio de tres sensores avanzados que no forman parte del pack de LEGO pero que se han utilizado en este trabajo. Finalmente se ha realizado un estudio de los distintos métodos de programación que se han desarrollado para este pack, las ventajas y desventajas que ofrece cada uno y una explicación más a fondo del software utilizado en este proyecto.

La segunda parte está dividida en las cuatro fases de aprendizaje que se han observado trabajando con el robot: proyectos de iniciación, bloques para sensores avanzados, proyectos avanzados y finalmente el diseño y montajes hardware. La primera fase consiste en la familiarización con la herramienta LEGO y su programación. La segunda fase estudia la aplicación de los sensores avanzados en la creación de nuevos proyectos. La tercera fase reúne lo aprendido durante las dos fases anteriores y lo utiliza para la creación de un proyecto de mayor dificultad. Por último, en la cuarta fase se estudia en profundidad la comunicación y la conexión entre el “ladrillo” y los sensores y se crea un nuevo sensor y actuador para el robot.

Las cuatro fases incluyen los ejemplos y montajes realizados, además de una reflexión final sobre los conocimientos aplicados o aprendidos durante la realización de los mismos.

Title: LEGO Mindstorms NXT as a learning tool in the engineering field

Author: Sandra Ramil Gomez

Director: Marcos Quílez Figuerola

Date: October, 9th 2009

Overview

The aim of this project is studying the LEGO Mindstorms NXT kit and evaluating its possible use as a learning tool for engineering studies. The project has been divided into two halves. The first half is the study of the tool itself. The second half presents diverse activities, of increasing difficulty, which allow the LEGO NXT to be used in the engineering studies that use a methodology of problem-based learning (PBL).

In the first half, a study has been carried out for each one of the pieces in the LEGO's pack: the brick, the motors and the sensors. In addition, three advanced sensors which do not form part of the standard LEGO pack have been analyzed because they will be used later. Finally a study has been carried out on the different programming methods developed for this pack, analyzing each one's advantages and disadvantages, followed by a more detailed explanation about the software used in this project.

The second half is divided into the four learning phases that have been observed while working with the robot: Novice projects, blocks for advanced sensors, advanced projects and finally the design and hardware assemblies. The first phase consists in familiarizing with the LEGO tool and its programming. The second phase studies the application of the advanced sensors in the creation of new projects. The third phase uses what has been learned during the previous phases and uses those concepts for the creation of a more difficult project. Finally, in the fourth phase the communication and the connection between the brick and the sensors is studied in depth and a new sensor is created.

The four phases include examples and the assemblies that have been done, plus a final overview on the knowledge applied or learned during the building of those assemblies.

Índice

Índice de figuras

Índice de tablas

INTRODUCCIÓN

Desde hace algunos años, profesores e investigadores utilizan la Robótica Pedagógica como herramienta de aprendizaje para sus alumnos. Sin ir más lejos, hace algunos meses en esta universidad se realizó un curso llamado “Robots, autòmats i d’altres enginys per ensenyar ciència i tecnología” cuyo objetivo era enseñar a profesores de secundaria los conocimientos necesarios para plantear proyectos didácticos a sus alumnos con LEGO Mindstorms NXT.

El concepto de Robótica Pedagógica consiste en la creación y utilización de robots físicos que permiten a los alumnos experimentar y aplicar conceptos de matemáticas, física, electrónica, informática y de las telecomunicaciones. Se utiliza tanto en la enseñanza primaria y secundaria como en las universidades y en la enseñanza para adultos de formación profesional. En la enseñanza primaria y secundaria, se utiliza para aprender principalmente los conceptos de programación y de matemáticas, apoyándose en la idea de “aprender jugando”. En las universidades se utiliza principalmente como herramienta de laboratorio, ya que permite practicar los conceptos aprendidos en las aulas.

Es aquí donde nace la idea de este trabajo, cuyo objetivo principal es investigar el uso de la robótica como herramienta de aprendizaje en la ingeniería. Es decir, desde mi punto de vista como estudiante de ingeniería técnica de telecomunicaciones a punto de finalizar la carrera, mi trabajo consiste en explorar el uso de los robots aplicando los conocimientos que he adquirido estos últimos 3 años. De esta forma, y si la experiencia es favorable, en adelante se podrá proponer el uso de robots como herramienta de soporte al aprendizaje de esos conocimientos en las mismas aulas en que los aprendí.

A medida que se extiende el concepto de Robótica Pedagógica, diversas industrias han desarrollado y puesto a la venta diversos kits para la construcción de robots: la empresa Minirobots con su Moway, RoboBuilder y su kit RoboBuilder modular 5710K, LEGO con su pack Mindstorms NXT,…

Para la realización de este trabajo, hemos escogido la herramienta LEGO Mindstorms NXT porque, a diferencia de otros productos como Moway o RoboBuilder, LEGO dispone de una gran versatilidad, puesto que permite el montaje y la creación de infinitos robots distintos solo combinando las diferentes piezas de LEGO y los sensores y motores de que dispone. Además, se trata de una herramienta potente, ya que el cerebro del robot, el ladrillo (o *brick,* como suele conocerse entre los aficionados al LEGO de habla inglesa) está formado por un microprocesador de 32-bit, memoria FLASH, un enlace

bluetooth, un puerto USB, y diversos puertos de entrada y salida que permiten la conexión y uso de motores y sensores de una forma muy sencilla.

El primer capítulo de este trabajo recoge un estudio de las características del pack de LEGO Mindstorms NXT: el ladrillo, los motores y los sensores. También se incluye en el estudio una serie de sensores avanzados que, aunque no forman parte del pack de LEGO, hemos adquirido para la realización de este trabajo. El último apartado del capítulo es un estudio sobre los entornos de programación utilizados en este proyecto y del resto de posibles entornos creados para LEGO Mindstorms NXT.

El segundo capítulo describe cuatro fases de aprendizaje con el kit NXT que se han observado en este proyecto:

- La primera de ellas consiste en la familiarización con LEGO, una serie de primeros proyectos sencillos que permiten aprender a utilizar nuestra herramienta, entender la programación, el montaje y la conexión con sensores y motores de nuestro robot.
- La segunda supone un paso más de dificultad y utiliza los sensores avanzados para crear robots. La dificultad principal de estos robots proviene de la necesidad de entender los datos que nos facilita cada sensor y como debemos tratarlos para nuestro fin.
- En la tercera parte creamos un proyecto nuevo y avanzado, que contiene alguno de los sensores o motores del pack más alguno de los sensores avanzados. Además, en este proyecto utilizamos 2 ladrillos que interactúan entre ellos mediante su conexión bluetooth.
- Por último, en la cuarta parte estudiamos el funcionamiento interno del ladrillo NXT y el método de conexión con sus sensores y actuadores. Con esos conocimientos, creamos y programamos nuestro propio sensor y actuador.

Todos los aparatos utilizados durante la realización de este trabajo cumplen la directiva 2050/95/CE de Restricción de ciertas Sustancias Peligrosas en aparatos eléctricos y electrónicos (RoHS)**.**

CAPÍTULO 1. Estudio y análisis del LEGO Mindstorms NXT como herramienta de aprendizaje.

1.1. NXT

El cerebro del Lego Mindstorms NXT es un módulo basado en un microcontrolador que ofrece diversos puertos de entrada y salida. Este módulo se conoce habitualmente como “el ladrillo”. Este ladrillo inteligente permite al robot cobrar vida y realizar diferentes operaciones. En algunos ámbitos, el ladrillo recibe el nombre de unidad NXT, o simplemente NXT. En este trabajo nos referiremos indistintamente a ella como “la NXT” o “el ladrillo”, como hacen los aficionados al LEGO NXT.

La NXT tiene 3 puertos de salida para conectar los motores (Puertos A, B y C), 4 puertos de entrada para conectar los sensores (Puertos 1, 2, 3 y 4) y un puerto USB para conectar el cable USB y descargar los programas desde el ordenador a la NXT (o bien cargar datos del robot al ordenador). También se pueden cargar y descargar los programas mediante la conexión Bluetooth.

Además, la NXT dispone de un altavoz que permite crear programas que reproducen sonidos reales.

Los botones de la NXT también pueden utilizarse como entradas en un programa. Dispone de 4 botones:

- botón de Enter (o botón naranja): On/Enter
- botón flecha izquierda y botón flecha derecha: Permiten desplazarse por el menú NXT
- botón gris inferior: Borrar/Volver atrás.

Por último, la pantalla de la NXT nos permite mostrar texto, dibujos predeterminados o bien creados por el usuario.

Especificaciones técnicas

- Microcontrolador ARM7 de 32-bits
- 256 Kbytes de memoria FLASH, 64 Kbytes de memoria RAM
- Microcontrolador AVR de 8-bits
- 4 Kbytes de memoria FLASH, 512 Kbytes de memoria RAM
- Comunicación inalámbrica Bluetooth (Bluetooth Clase II V2.0 compliant)
- USB full speed port (12 Mbit/s)
- 4 puertos de entrada

- 3 puertos de salida
- LCD pantalla gráfica de 100 x 64 pixeles
- Altavoz de calidad de sonido 8 KHz. Canal de sonido con 8 bits de resolución y un sample rate de 2 a 16 KHz.
- Alimentación: 6 pilas AA / Batería recargable de litio

1.2. Motores

Los tres servo motores proporcionan al robot la habilidad de moverse. Cada motor dispone de un sensor de rotación que permite controlar los movimientos del robot con precisión. El sensor de rotación mide las rotaciones del motor en grados o rotaciones completas (con una exactitud de +/- un grado). Una rotación es igual a 360 grados. El sensor de rotación también permite fijar distintas velocidades al motor, cambiando el parámetro potencia (power).

1.3. Sensores básicos

El set NXT incluye 4 sensores diferentes: el sensor de tacto, el sensor de sonido, el sensor de luz y el sensor de ultrasonidos. A continuación, se estudiara cada uno de los sensores en profundidad y algunas de sus aplicaciones.

1.3.1. Sensor de contacto

El sensor de tacto le proporciona al robot el sentido del tacto. Es capaz de detectar cuando está siendo presionado por algo y cuando vuelve a su posición inicial. Por tanto, podemos distinguir entre tres situaciones distintas:

- Presionar el botón
- Dejar de presionar el botón
- Presionar y soltar el botón

Este sensor dispone además de un hueco en forma de cruz que permite conectarlo directamente a otros montajes. Internamente, este sensor está formado por un interruptor y un conector.

Se podría utilizar este sensor para construir un robot capaz de coger cosas: el sensor estaría incluido en el brazo del robot y detectaría cuando hay algo que coger. Otro uso bastante común sería el de detector de choque en un robot

vehículo. El robot-vehículo podría decidir su propia dirección y cambiarla en caso de detectar presión sobre el botón, lo cual significaría que hemos chocado con algo.

1.3.2. Sensor de sonido

El sensor de sonido le proporciona al robot el sentido del oído. Es capaz de expresar la medida tanto decibelios [dB] como decibelios con ajuste A [dBA].

dBA: En la detección de decibelios con ajuste A, la sensibilidad del sensor está ponderada por la sensibilidad del oído humano. Es decir, estos son los sonidos que puede escuchar una persona.
dB: En la detección de decibelios estándar, todos los sonidos se miden con la misma sensibilidad. Por tanto, estos sonidos incluirán algunos que serán demasiado altos o demasiado bajos para el oído humano.

El sensor de sonido puede medir niveles de la presión del sonido hasta 90 dB, aproximadamente el nivel de un cortacésped. Los niveles de presión del sonido son extremadamente complicados y, por ello, las lecturas del sensor de sonido se muestran en tanto por ciento [%]. A menor porcentaje, más silencioso es el sonido. Por ejemplo:

- 4-5% es una habitación silenciosa
- 5-10% sería alguien hablando a cierta distancia
- 10-30% es una conversación normal cerca del sensor o música sonando con el volumen normal
- 30-100% es gente gritando o música sonando con el volumen muy alto

Como ejemplo de aplicación simple, este sensor se podría utilizar para realizar un robot contador de palmadas.

1.3.3. Sensor de Luz

El sensor de luz es uno de los dos sensores que proporciona al robot el sentido de la visión (el otro es el sensor de ultrasonidos). El sensor de luz permite al robot distinguir entre la luz y la oscuridad. Puede leer la intensidad de luz en una habitación y medir la intensidad de luz de la superficies coloreadas.
En la **Fig 1.1** se muestra lo que el ojo humano percibe:

Fig 1.1 Percepción de un ojo humano.

Y en la **Fig 1.2** se muestra lo que vería el robot, utilizando el sensor de luz.

Fig 1.2 Percepción del sensor de luz.

Este sensor dispone de un LED que puede activarse mediante software, de tal manera que permite medir la luz reflejada sobre un objeto, o simplemente la luz ambiental.

El fototransistor del sensor es bastante más sensible a los colores infrarojos de la luz que a la pequeña parte del espectro visible que vemos. La **Fig 1.3** muestra la respuesta espectral del transistor junto a la del ojo humano:

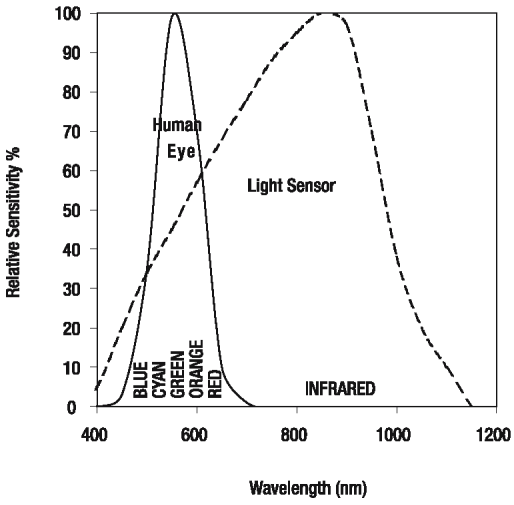

Fig 1.3 Respuesta frecuencial del sensor de luz.

Tal como se observa en la figura anterior, mientras el ojo humano diferencia más fácilmente el verde del azul o del rojo, el sensor distingue con mayor

facilidad el rojo del azul. Seguramente, esta sea la razón por la que LEGO ha escogido esos colores para las dos bolas incluidas en su set.

Se podría utilizar este sensor para construir un robot que siga líneas, o que decida entre un objeto u otro según el color de este.
También se podría utilizar como robot alarma, que reaccionase cuando alguien encendiese la luz de la habitación.

1.3.4. Sensor de Ultrasonidos

El sensor de ultrasonidos es el otro sensor capaz de proporcionar el sentido de la vista al robot. Este sensor permite al robot ver y detectar objetos. También puede usarse para crear un robot capaz de sortear objetos, medir distancias y detectar movimientos.

Internamente, este sensor está formado por dos transductores de ultrasonidos, un emisor y un receptor. Se trata de un sensor complejo que requiere de su propio microprocesador. El sensor trabaja como un sonar, enviando un pulso de ultrasonido de 40kHz y midiendo el tiempo que tarda el sonido en viajar hacia un objeto, reflejarse y volver.

El sensor de ultrasonidos mide distancias en centímetros y en pulgadas. Permite medir distancias de entre 0 y 255 cm con una precisión de +/- 3 cm calculando el tiempo que tarda una onda sonora en chocar con un objeto y volver, como un eco.

Dependiendo de la forma y el material de que este hecho el objeto, será más fácil o no detectar el objeto. Por ejemplo, si el objeto es muy grande y de superficie dura devolverá una lectura muy buena, mientras que un objeto curvado o muy fino será más difícil de detectar por el sensor.

1.4. Sensores avanzados

Algunos fabricantes han creado una nueva gama de sensores para LEGO Mindstorms NXT. La mayoría de estos sensores utilizan el mismo encapsulado y están certificados por la compañía de LEGO, demostrando así lo siguiente:

- Son 100% compatibles con Mindstorms NXT
- Cumplen los estándares de calidad de LEGO
- Cumple los estándares de seguridad
- Son conformes a la directiva europea de residuos (RoHs)

HiTechnic es el fabricante que más sensores ha creado para LEGO Mindstorms NXT: el acelerómetro o sensor de inclinación, el sensor de color, la brújula, el detector electro-óptico de proximidad, el giróscopo y el sensor de enlaces por infrarrojos, etc. Además, Hitechnic ofrece un Prototipo de placa NXT para crear con ella tus propios sensores para Mindstorms NXT.

Para este trabajo, vamos a estudiar y utilizar los siguientes sensores: el giróscopo, el acelerómetro y la brújula.

1.4.1. Giróscopo

Este Sensor contiene un sensor giroscópico de un solo eje que detecta la rotación y devuelve la velocidad de rotación en grados por segundo. Puede medir hasta un máximo de +/-360º por segundo de rotación.

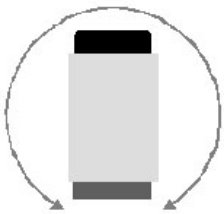

Fig 1.4 Posición del eje de medida en el sensor giróscopo.

Este sensor se conecta a uno de los cuatro puertos de la NXT destinados a los sensores utilizando un cable estándar de la NXT y utiliza la interface de sensor analógico. La ratio de rotación se puede leer hasta un máximo de 300 veces por segundo.

El eje de medida del sensor se encuentra en el plano vertical, con el giróscopo posicionado con la pieza negra final mirando hacia arriba, como se muestra en la **Fig 1.4**.

1.4.2. Acelerómetro

Este Sensor contiene un acelerómetro que mide la aceleración en tres ejes, X, Y y Z. La aceleración se mide en un rango de -2g a +2g con una escala de aproximadamente 200 cuentas por g.

El sensor Acelerómetro también se puede utilizar para medir la inclinación en los tres ejes.

Este sensor se conecta a uno de los cuatro puertos de la NXT destinados a los sensores utilizando un cable estándar de la NXT y utiliza el protocolo de comunicaciones digital I^2C. Las medidas de la aceleración para cada eje se refrescan aproximadamente 100 veces por segundo.

Los tres ejes de medida están etiquetados tal y como se muestra en la **Fig 1.5**.

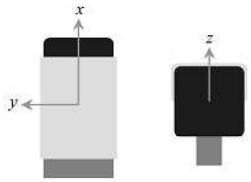

Fig 1.5 Ejes de medida en el sensor acelerómetro.

En la **Tabla 1.1** podemos observar de qué forma está ordenado el buffer de datos que obtenemos del sensor.

Tabla 1.1 Diseño del registro del sensor acelerómetro.

Dirección	Tipo	Contenido
42H	byte	8 bits más altos del eje X
43H	byte	8 bits más altos del eje Y
44H	byte	8 bits más altos del eje Z
45H	byte	2 bits más bajos del eje X
46H	byte	2 bits más bajos del eje Y
47H	byte	2 bits más bajos del eje Z

1.4.3. Brújula

Este Sensor contiene una brújula magnética digital que mide el campo magnético de la tierra y calcula un ángulo respecto al norte magnético. La sensibilidad del sensor es de un grado y trabaja a una frecuencia de 100Hz. La Brújula se conecta a uno de los cuatro puertos de la NXT destinados a los sensores utilizando un cable estándar de la NXT.

1.5. Entornos de programación

LEGO Mindstorms permite una gran variedad de entornos de programación: NXT-G, Labview, RoboC, PBLua, NBC-NXC, Lejos,… Debido al límite de tiempo y teniendo en cuenta que nuestro objetivo no es profundizar en la programación, en este trabajo solo hemos utilizado NXT-G y Labview. Sobre estos dos entornos de programación daremos una explicación más detallada y, finalmente, realizaremos una pequeña comparativa. No obstante, también realizaremos un breve resumen de las características del resto de entornos de programación.

1.5.1. NXT-G

NXT-G es el software proporcionado por LEGO Mindstorms que permite programar tus propios robots creados con el kit y descargar los programas a la NXT mediante conexión USB o Bluetooth.

Este software está basado en el motor de Labview de National Instruments y, como Labview, utiliza el llamado Lenguaje G. Este lenguaje describe de forma visual los elementos y el funcionamiento del programa sumándole simplicidad a su elaboración. Se trata de una programación mediante bloques que permiten programar un robot sencillo rápidamente. Crear un programa con NXT-G es muy parecido a crear un organigrama. Cada uno de los bloques de programación permite al robot realizar un movimiento (controla los motores) o bien recibir datos de un sensor. En la **Fig 1.6** se muestra la pantalla principal del programa.

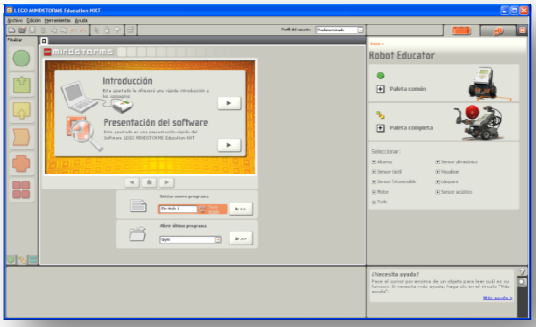

Fig 1.6 Pantalla inicial del programa NXT-G.

El software dispone de 3 paletas: común, completa y hecha a medida. La paleta común dispone únicamente de 7 sencillos bloques: movimiento, grabación/reproducción, sonido, display, espera, bucle y condición. La paleta completa agrupa los bloques en categorías según su función: lectura de sensores, puesta en marcha de actuadores (motores y altavoz), control del flujo del programa, etc. También incluye un grupo de bloques para la utilización de los datos; comparar, sumar, restar, multiplicar, dividir, etc. Finalmente también permite crear variables locales numéricas, lógicas o textuales. Esta paleta es la más útil, ya que permite crear todo tipo de robots e incluso que interactúen entre ellos, ya que dispone de los bloques para la comunicación bluetooth. La paleta hecha a medida nos permite crear o descargar nuestros propios bloques, ya sea para agrupar un conjunto de acciones que se repite en nuestro programa o bien para utilizar un nuevo sensor que hayamos adquirido.

Este software es el más intuitivo y fácil de utilizar, ya que los bloques están creados específicamente para la NXT e incluso dispone de una gran variedad de ejemplos sobre robots para que cualquiera pueda seguir las instrucciones y crear su propio robot.

Sin embargo, llegados a cierto nivel del proyecto en que la programación de nuestros robots era elevada, nos dimos cuenta de algunas de las limitaciones de este software:

- Es poco manejable, puesto que la única posibilidad de que dispone para moverse por la pantalla es la herramienta mano, y no permite alejarse o acercarse según necesidad.
- Las funciones de programación son limitadas, sobre todo respecto a Labview, como se verá posteriormente.
- La conexión con la NXT por bluetooth no funciona correctamente y acostumbra a fallar.

Por tanto, este software es una buena herramienta para familiarizarse con la NXT y dar los primeros pasos con nuestros robots, pero no es recomendable para una programación avanzada.

1.5.2. Labview

Labview es un poderoso entorno de desarrollo gráfico con funciones integradas para realizar adquisición de datos, control de instrumentos, análisis de medida y presentaciones de datos. Como ya se ha dicho anteriormente, este entorno utiliza el Lenguaje G, de tal forma que es posible escribir programas altamente complejos con una interfaz de usuario completa y a medida.

Los programas desarrollados con Labview se llaman Instrumentos Virtuales, o VIs, y se dividen en dos partes: Panel Frontal y Diagrama de Bloques. El Panel Frontal es la interfaz del usuario, en la cual se definen los controles e indicadores que se muestran en pantalla. El Diagrama de Bloques es el programa propiamente dicho, donde se define su funcionalidad, aquí se colocan los bloques que realizan una determinada función y se interconectan.

Labview dispone de un **Toolkit para LEGO Mindstorms NXT** que permite utilizar las herramientas avanzadas de programación grafica que Labview provee al usuario para el control del NXT escapando de las limitaciones propias del NTX-G. Ofrece más libertad de control y expande los límites de lo que es posible en el desarrollo de proyectos más complejos. Al ser un entorno ampliamente usado en ámbitos profesionales y sectores académicos encaja perfectamente.

Utilizar el toolkit proporcionado por NI para la NXT agrega tres grandes conjuntos de VIs que abren las posibilidades al usuario para que pueda:

- Compilar y descargar un programa elaborado con Labview al NXT pudiendo interactuar con este mientras el programa está en ejecución. Agregando un control, el toolkit puede enviar datos al modelo e influenciar el comportamiento del programa en ejecución. Agregando un indicador, el toolkit envía de regreso un valor en ese punto del programa a la PC para que el usuario pueda observarlo y obtener actualizaciones en tiempo real del NXT durante la operación de un programa por medio de los paneles frontales de Labview.
- Escribir un programa en Labview que funcionará en el PC y se comunique con la NXT a través del USB o el Bluetooth.
- Si el usuario es desarrollador de un nuevo sensor o de nuevos componentes de hardware Labview le permitirá crear bloques nativos para la programación y control del hardware creado para su uso en el entorno propio del MINDSTORMS (es necesario usar el toolkit con la versión 7.1 de Labview por compatibilidad ya que el NXT-G está basado en esta versión).

Además de las nombradas anteriormente, Labview también nos permite una fácil conexión con la NXT, ya sea vía USB o vía Bluetooth. Sin embargo, y sabiendo que Labview es un programa que dispone de infinitas funciones y que no se ha creado exclusivamente para NXT, Labview no permite descargar a la NXT todos los programas que puedas crear, puesto que dispone de funciones que no es capaz de transformar a código bit.

La **Tabla 1.2** es un comparativo resumen entre las dos herramientas de programación explicadas anteriormente.

Tabla 1.2 Comparativo NXT-G y Labview.

NXT-G		Labview	
Ventajas	**Inconvenientes**	**Ventajas**	**Inconvenientes**
Fácil para programadores inexpertos	Dificultad al crear programas complejos (poco manejable)	Programación avanzada sin demasiada complejidad	No es capaz de convertir a código bit todos los programas creados.
Permite compilar y convertir a código bit todos los programas creados.	Difícil conexión por Bluetooth	Manejable	
	Funciones de programación limitadas	Fácil conexión por Bluetooth y USB	
		Permite manejar la NXT desde el PC sin necesidad de descargar el programa en ella	

1.5.3. Lenguajes de programación no oficiales para la NXT

Para aquellos que prefieren escribir líneas de código a colocar y conectar bloques, existen una serie de lenguajes de programación no oficiales para la NXT. A continuación vamos a introducir cuatro de estos lenguajes: NBC; NXC, lejos NXJ y RobotC.

Next Byte Code (NBC)

Este lenguaje, desarrollado por John Hansen, fue el primer lenguaje de programación textual para la NXT y está basado en el lenguaje ensamblador. El código es sencillo, ya que consiste principalmente en palabras o abreviaturas. Además, utiliza el firmware estándar de la NXT y el software se puede descargar gratuitamente en internet.

Not eXactly C (NXC)

NXC es un lenguaje de programación de alto nivel desarrollado por Hansen poco después de crear NBC. Tal y como su nombre indica, NXC está basado

en el lenguaje de programación C. Como en el caso de NBC, el software se puede descargar gratuitamente y comparte página web con el anterior.
Estos dos lenguajes pueden ser útiles para usuarios que no tengan experiencia en programación, ya que no son complicados de entender, aunque también aportan rasgos que satisfacen a los programadores avanzados.

leJOS NXJ

Lejos NXJ es un leguaje de programación para LEGO NXT basado en Java. Se puede obtener gratuitamente la última versión de este lenguaje desde la web de leJOS. Sin embargo, a diferencia de NBC y NXC, lejos NXJ requiere la utilización de un firmware propio en la NXT llamado Java Virtual Machine (JVM). Además, se necesitan conocimientos de Java para poder utilizar este lenguaje correctamente. Por tanto, se trata de un lenguaje bastante más avanzado y complicado de utilizar que NBC o NXC.

RobotC

RobotC es un producto comercial que intenta captar el mercado educacional y soporta algunas microcomputadoras además de la NXT. Este lenguaje está implementado en C y requiere de la utilización de su propio firmware en la NXT. Las licencias se pueden conseguir por internet, aunque también se puede probar el software durante 30 días de forma gratuita.

El firmware de RobotC tiene numerosas mejoras respecto al firmware estándar, ya que, por ejemplo, ejecuta el código más rápidamente, realiza una mejor gestión de la memoria y tiene mejores capacidades respecto al sonido.
RobotC es adecuado tanto para programadores principiantes como para expertos.

CAPÍTULO 2. Aplicaciones del LEGO Mindstorms NXT como herramienta de aprendizaje

2.1. Proyectos de iniciación

Con el pack de LEGO Mindstorms NXT y el software proporcionado por el fabricante, cualquiera debería ser capaz de construir su propio robot de LEGO. El propio software, NXT-G, dispone de un apartado llamado Robo Center en el cual podemos encontrar un listado de proyectos agrupados según el tipo de robot de que se trata (vehículos, maquinas, animales o humanoides) con una explicación paso a paso del montaje del robot y de su programación. Además, se pueden encontrar una gran variedad de proyectos sencillos colgados en internet de otros aficionados a LEGO Mindstorms NXT. Para entender el funcionamiento del robot y la utilidad de sus sensores, así como del lenguaje de programación de este software, es un buen paso de iniciación revisar y construir algunos de los proyectos facilitados por NXT-G.

El primer robot sencillo que vamos a crear es un robot alarma que utiliza el sensor de sonido para detectar ruidos dentro de una casa vacía. El montaje (**Fig 2.1**) consiste únicamente en nuestro ladrillo NXT unido al sensor de sonido. De hecho, nos interesa que sea un montaje pequeño ya que necesitaremos esconderlo.

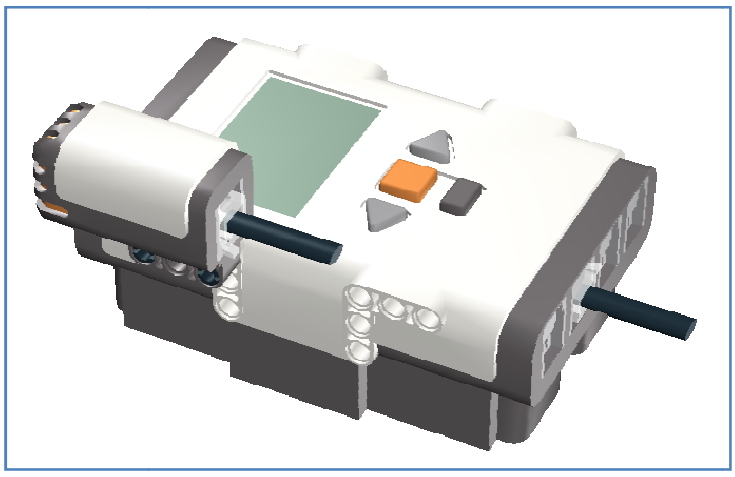

Fig 2.1 Montaje Robot-Alarma.

El robot se mantendrá a la espera de recibir una medida del sensor de sonido mayor del 10% (valor correspondiente a una conversación o a música sonando) mientras en pantalla se observa la imagen de un candado cerrado. Si se

detecta un valor del sensor de sonido mayor al 10%, espera durante 10 segundos a que alguien presione el botón de Enter (señal de apagado). En caso de que alguien lo presione, aparece en pantalla un candado abierto y guardamos el valor “true” en la variable lógica que indicará el fin del bucle principal y, por tanto, del programa. En caso de que nadie lo presione, sonará una alarma que dice “Attention” y no parará hasta que alguien apague el programa. En la **Fig 2.2** se observa el diagrama de bloques del programa.

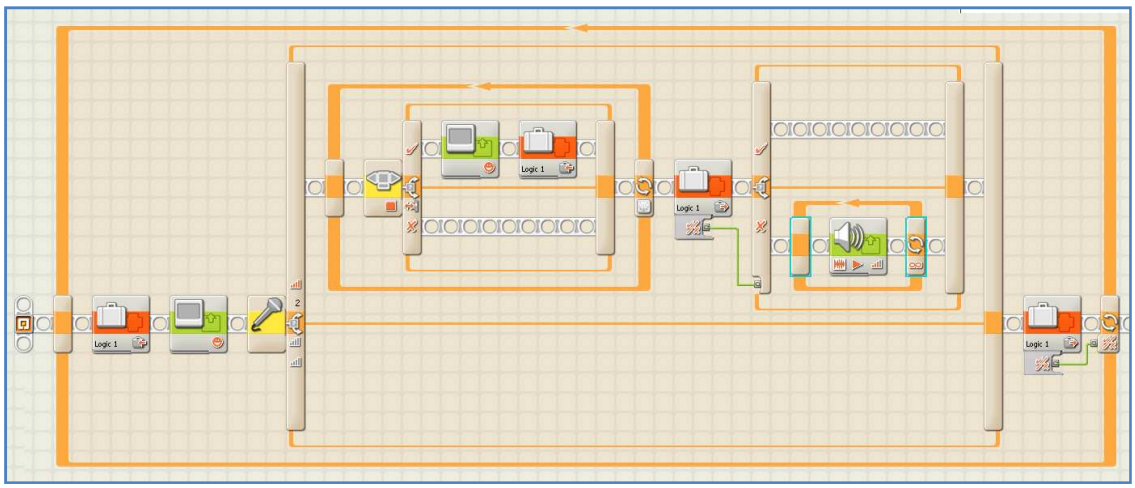

Fig 2.2 Diagrama de bloques del programa Robot-Alarma.

El siguiente robot sencillo que vamos a crear es un robot andador que utiliza los dos sensores capaces de proporcionarle vista al robot (el sensor de tacto y el sensor de ultrasonidos) para evitar obstáculos. El montaje que utilizaremos será el diseño de vehículo simple que nos ofrece LEGO en su guía de iniciación (**Fig 2.3**).

Fig 2.3 Montaje del Robot-vehículo.

La programación para este robot es la siguiente: el robot se desplaza hacia adelante siempre que el sensor de ultrasonidos no detecte ningún obstáculo a menos de 20cm de distancia. Si lo detecta, el robot gira hacia la derecha y empieza a andar marcha atrás hasta que el sensor de tacto detecta un choque. Entonces gira hacia la izquierda y se repite de nuevo el bucle. En la Fig 2.2 se observa el diagrama de bloques del programa.

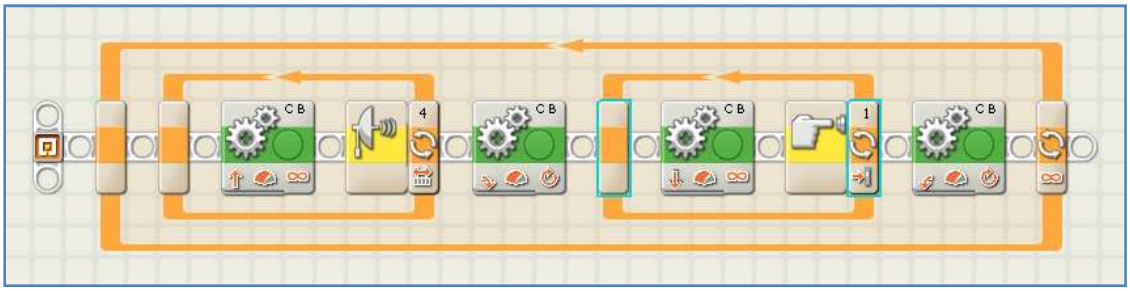

Fig 2.4 Diagrama de bloques del programa Robot-vehículo.

Finalmente vamos a crear un robot-vehículo que detecte su objetivo gracias a unos emisores de luz infrarroja. Es decir, cada emisor de infrarrojos emitirá luz con una determinada duración y el robot-vehículo deberá detectar ese emisor y la duración de su luz para saber si se trata del objetivo correcto. En caso de que el emisor detectado no coincida con los datos del emisor correcto, el robot-vehículo seguirá dando vueltas hasta encontrar el siguiente emisor. La **Fig 2.5** muestra el montaje utilizado.

Fig 2.5 Montaje del robot detector de infrarrojos.

La programación para este robot es la siguiente: Inicializamos dos variables que serán las encargadas de contar dos intervalos del tiempo en que la luz infrarroja está encendida. El robot empieza a dar vueltas hasta que detecta luz. En ese instante para y se pone en marcha el primer contador (timer). Cuando deja de detectar luz el sensor, guardamos el valor del temporizador 1 en la variable time (**Fig 2.6**).

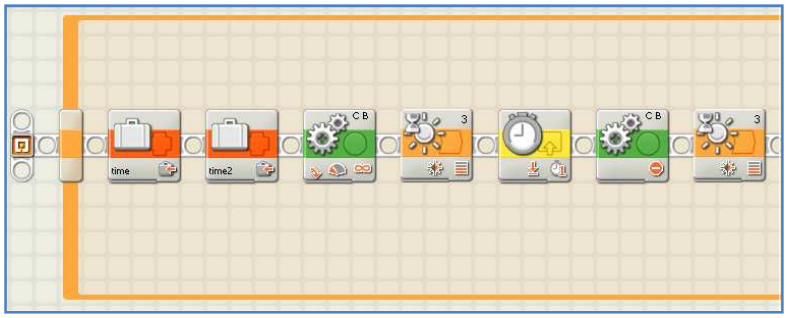

Fig 2.6 Diagrama de bloques del robot detector de infrarrojos, parte 1.

Acto seguido esperamos a volver a detectar luz para iniciar el timer 2 y repetir la operación anterior para acabar guardando el valor del contador en la variable time 2. Nos aparece en pantalla el valor de los dos timers en milisegundos (**Fig 2.7**).

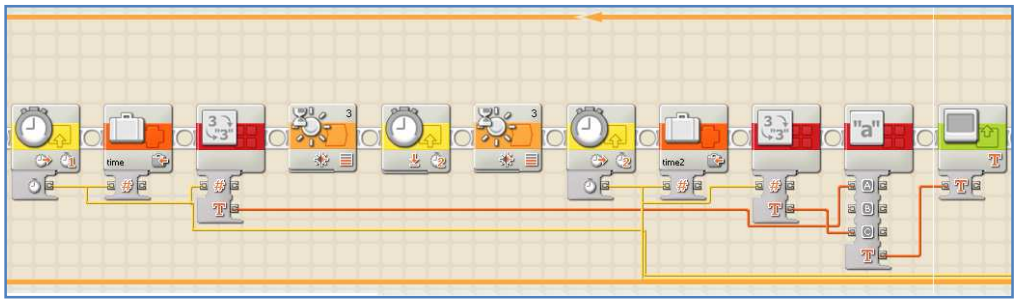

Fig 2.7 Diagrama de bloques del robot detector de infrarrojos, parte 2.

Finalmente comparamos si el valor de los timers coincide con el tiempo en que nuestra luz infrarroja se mantiene encendida y, si coincide, aparece un smile en la pantalla de la NXT y nos saluda con un “Hello”. En este caso, habríamos encontrado a nuestro objetivo (**Fig 2.8**).

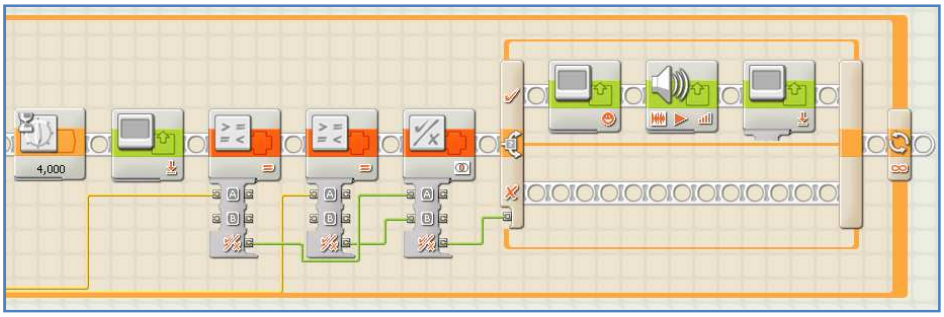

Fig 2.8 Diagrama de bloques del robot detector de infrarrojos, parte 3.

Si no coinciden los valores, vuelve a empezar el proceso de búsqueda.

Conocimientos adquiridos

Esta actividad permite:

- Aprender a utilizar correctamente el software NXT-G (programación con lenguaje G)
- Aprender a utilizar correctamente los sensores del pack Mindstorms NXT y la utilidad de los datos que nos ofrecen.

2.2. Bloques para sensores avanzados

Como se ha mencionado anteriormente, disponemos de una serie de sensores avanzados, es decir, sensores que LEGO no facilita en su pack NXT.

Para utilizar estos sensores, Hi-technic permite descargar los bloques para programar con Labview y con NXT-G. Sin embargo, la mayoría de los datos que obtenemos con estos bloques no son útiles sin un procesado posterior.

Empezaremos por el giróscopo que, como ya se ha dicho, es un sensor capaz de detectar la rotación en un solo eje y nos devuelve la velocidad de rotación en grados por segundo. Los giróscopos se utilizan en las seagway para detectar su velocidad de giro y ser capaz de contrarrestar su inclinación.

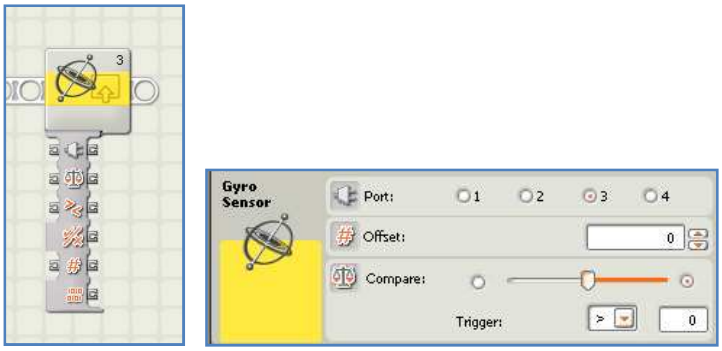

Fig 2.9 Bloque del sensor giróscopo para NXT-G.

El bloque para el sensor giróscopo (**Fig 2.9**) nos permite seleccionar el puerto en que conectaremos el sensor, adjudicar un nivel de offset, es decir, la medida que nos da el giróscopo en reposo y utilizar un trigger si lo que queremos es comparar la medida obtenida.

El primer programa que vamos a utilizar para probar el sensor giróscopo se llama Gyro Test (**Fig 2.10**) y es uno de los programas descargables que proporciona Hitechnic en su web. Este programa, realizado en NXT-G, hace una primera calibración tomando 10 medidas del sensor en reposo y haciendo la media. El resultado lo utilizará como offset del sensor. Finalmente, las medidas obtenidas con el sensor se mostrarán en la pantalla de la NXT.

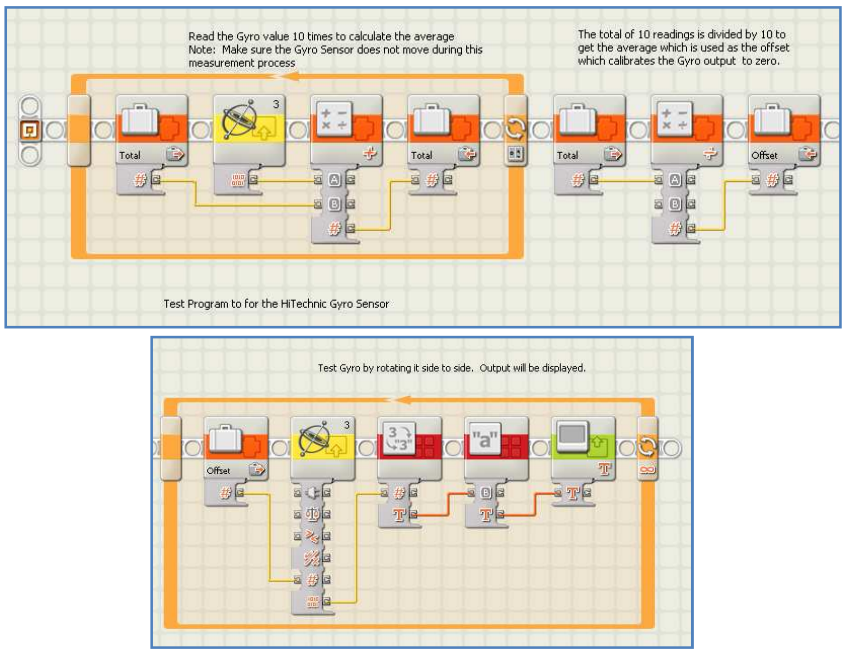

Fig 2.10 Diagrama de bloques del programa Gyro Test

De este programa hemos observado que el offset del giróscopo era de 608 unidades.

Tratando de buscar una utilidad a los datos que nos ofrecía el giróscopo con uno de nuestros robots sencillos, el montaje del robot andador, nos dimos cuenta de que el robot se desviaba ligeramente y no seguía una línea recta al caminar. Para ser capaz de corregir esa desviación cuando nos dirigíamos hacia un punto en concreto, sería interesante conocer al final de nuestro desplazamiento el número de grados que nos habíamos desviado de nuestro camino. Puesto que el giróscopo nos ofrece velocidad en grados por segundo y necesitamos saber los grados girados, la solución más fácil sería utilizar un contador que nos guardase el tiempo en segundos y multiplicar esa cifra por la velocidad marcada por el giróscopo. Sin embargo, esta solución solo podría aplicarse si la velocidad de giro fuese constante y, puesto que se trata de un robot en movimiento cuyo cambio de dirección puede depender de varios factores externos (el viento, un obstáculo en su camino, una persona que lo empuje, etc.) no es una solución válida.

Llegados a este punto, nos damos cuenta de que lo que necesitamos es crear un vector que guarde los datos que obtenemos del giróscopo e integrarlos según el intervalo de tiempo que tardamos en recibir un nuevo dato. Puesto que el software proporcionado para la NXT, NXT-G, no permite crear vectores y tampoco permite integrar, para este proyecto utilizaremos Labview.

El primer programa que vamos a crear, nos permitirá ver en pantalla dos gráficas, una mostrará la velocidad en grados por segundo obtenida por el giróscopo, y la otra mostrará los grados girados, resultado de integrar los datos de la gráfica anterior (**Fig 2.11**).

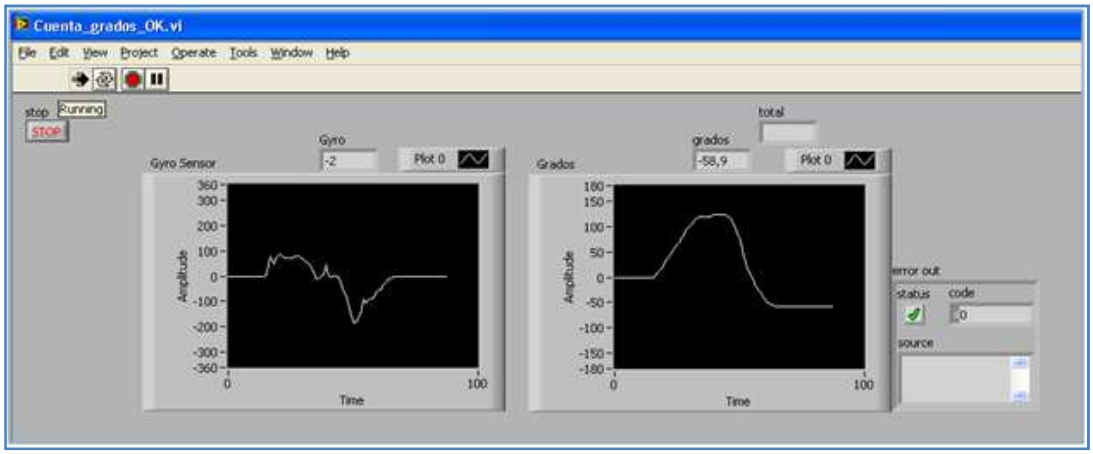

Fig 2.11 Interfaz de usuario del programa Cuenta grados.

Al apretar el botón de ‘stop’ en el indicador llamado ‘total’ aparecerá la suma de los grados girados des del inicio del programa. En la **Fig 2.12** se observa el diagrama de bloques del programa.

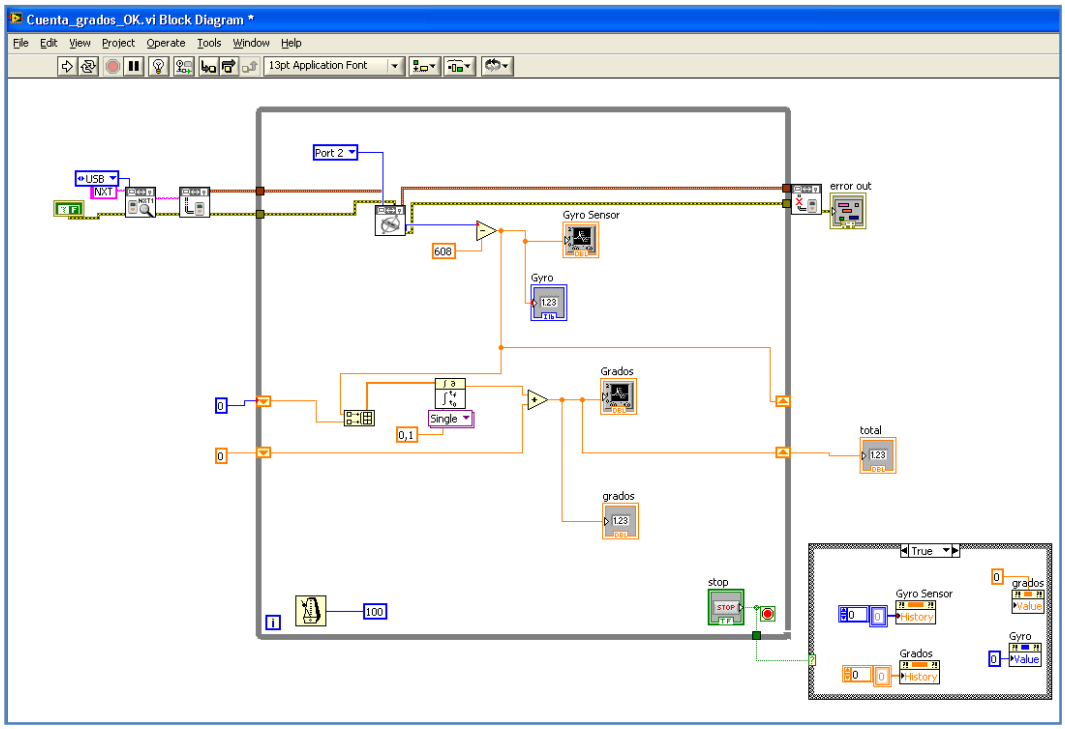

Fig 2.12 Diagrama de bloques del programa Cuenta grados.

Utilizando el mismo offset calculado en el programa Gyro Test, obtenemos la medida del giróscopo y la guardamos en un vector junto a la medida anterior (que al principio es 0) e integramos por intervalos de 100ms (tiempo que tardamos en realizar otra medida). Finalmente, en otra variable vamos sumando y guardando el numero de grados girados para obtener el total.

Al realizar la prueba del programa, nos damos cuenta de que, aun en estado de reposo, se detecta una velocidad de giro. Este error es debido a que hemos escogido una medida errónea del offset, ya que el valor de la medida del giróscopo en reposo no es estable y oscila entre dos valores (608 y 609). Puesto que no podemos añadirle un bit, la única solución que nos queda es restarle sensibilidad al sensor e inutilizar el último bit del integer de 16 de datos. De esta manera conseguimos estabilizar el valor de offset a 608 (**Fig 2.13**).

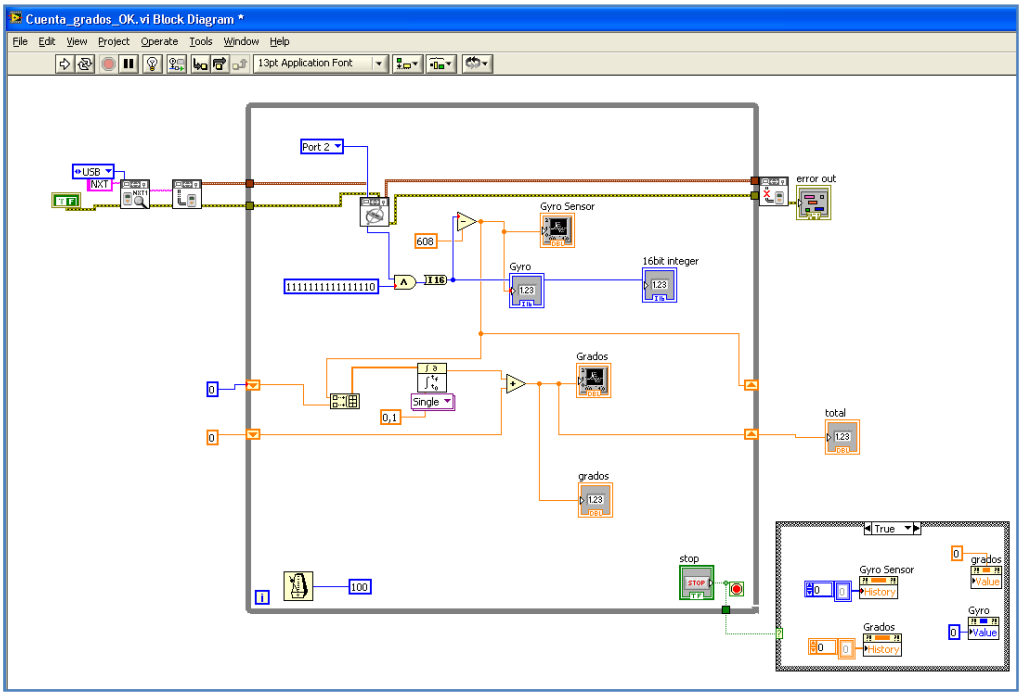

Fig 2.13 Diagrama de bloques del programa Cuenta grados corregido.

Esta modificación del programa funciona correctamente, pero, ya que se ha creado con la funcionalidad de Labview 'direct commands', es un programa que funciona sobre el PC y no sobre la NXT, lo cual limita bastante su utilidad si se trata de un robot en movimiento. Por tanto, vamos a adaptar nuestra solución a un programa descargable a la NXT (**Fig 2.14**). La diferencia entre este y el anterior reside principalmente en que no podemos utilizar el bloque integrador. A cambio, multiplicaremos la velocidad que nos devuelve el giróscopo por el intervalo de tiempo que definiremos para tomar las medidas. Sin embargo, hay que tener en cuenta que la NXT solo trabaja con números enteros. Por ello hemos escogido un intervalo de tiempo de 0,5 segundos entre cada medida, y resolveremos la operación anterior dividiendo por 2.

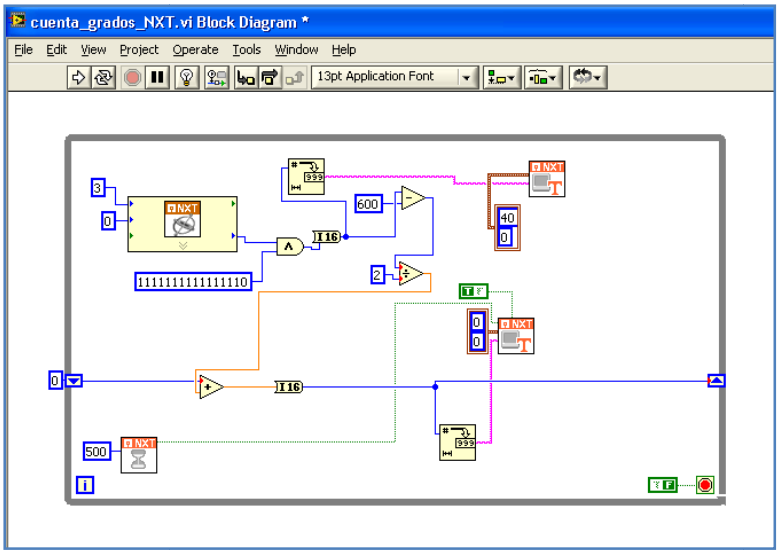

Fig 2.14 Diagrama de bloques del programa Cuenta grados para NXT.

Con ello realizamos una aplicación final utilizando un montaje muy simple, el ladrillo junto a un motor que a su vez tiene conectado en la rueda una pieza en la que colocamos el giróscopo (**Fig 2.15**). Se trata de que el programa sea capaz de deshacer los grados girados manualmente en ese motor. Para este caso, sería mucho más sencillo utilizar la capacidad de grabar y reproducir movimientos de que disponen los motores, pero más adelante, en el apartado **2.3 Proyectos avanzados** utilizaremos esta aplicación para un caso en que el giróscopo no gira sobre el rotor del motor.

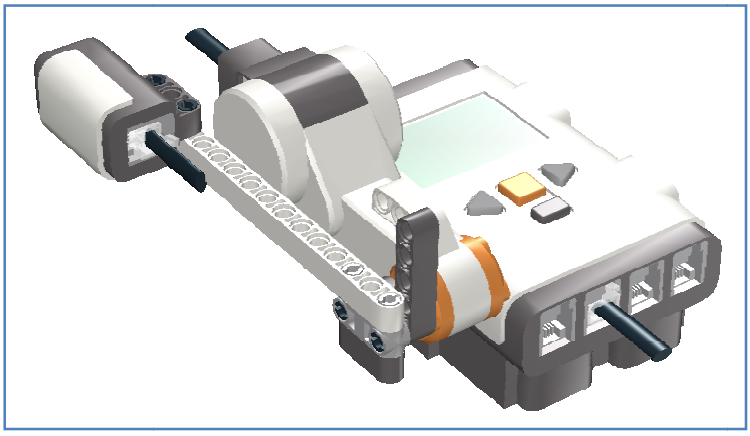

Fig 2.15 Montaje del programa Recupera grados.

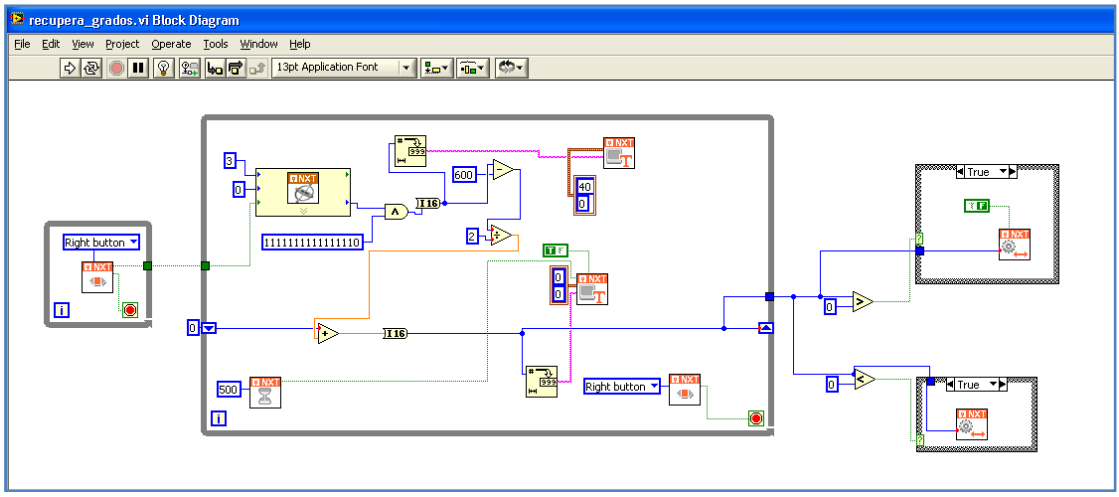

Fig 2.16 Diagrama de bloques del programa Recupera grados.

El programa (**Fig 2.16**) espera a que pulsemos el botón derecho del ladrillo y a continuación realiza la cuenta de los grados totales que giramos, tal y como lo hacíamos en el programa anterior, hasta que volvemos a apretar el botón derecho. Finalmente, recupera la posición inicial.

A continuación vamos a hablar del sensor acelerómetro. Este sensor, como ya se ha dicho anteriormente, mide la aceleración en los 3 ejes X,Y y Z.

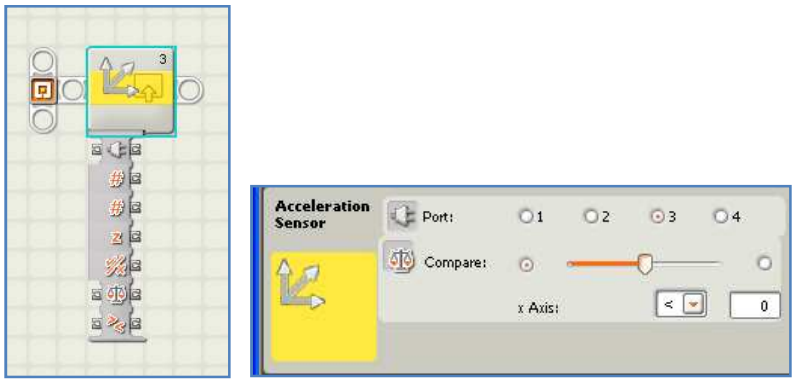

Fig 2.17 Bloque del sensor acelerómetro para NXT-G

El bloque del sensor acelerómetro (**Fig 2.17**) nos permite seleccionar el puerto en que conectaremos el sensor y comprobar si la medida obtenida es mayor o menor que el número introducido.
En primer lugar, vamos a crear un programa muy sencillo que nos permita ver en la pantalla de la NXT las medidas de los tres ejes (**Fig 2.18**).

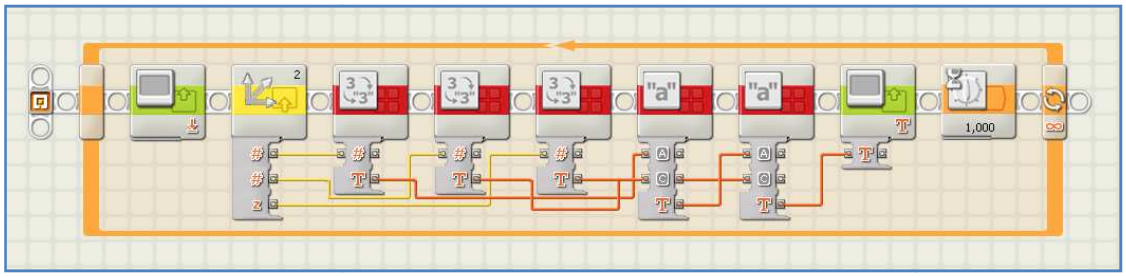

Fig 2.18 Diagrama de bloques del programa Accel test.

El montaje que vamos a utilizar consiste únicamente en el ladrillo conectado al sensor acelerómetro (**Fig 2.19**). A medida que movemos el ladrillo con el acelerómetro, observamos en pantalla las medidas obtenidas. Ya que sabemos que la gravedad se detecta con un valor de 200 unidades, somos capaces de descifrar la situación de cada eje, girando la NXT y dejándola en reposo.

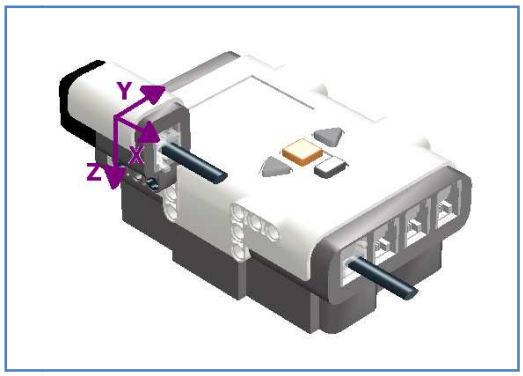

Fig 2.19 Montaje del programa Accel test.

Labview nos permite crear un programa igual al anterior pero con una interfaz de usuario mucho más atractiva, una gráfica para la medida de cada eje (**Fig 2.20**).

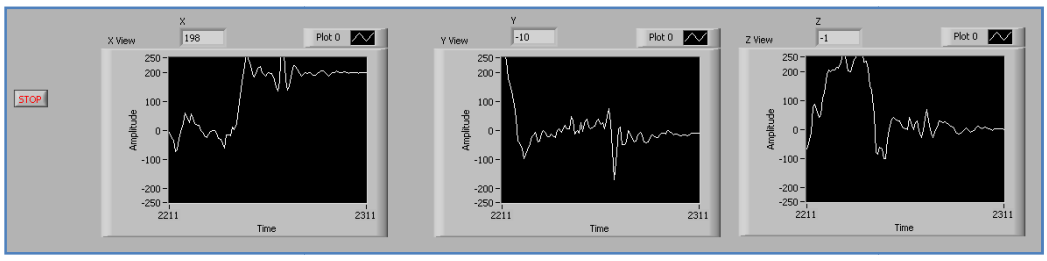

Fig 2.20 Interfaz de usuario del programa Accel test.

Realizando pruebas con este programa observamos que no siempre obteníamos una medida correcta de la aceleración en el eje Y y en el eje Z ya que, a diferencia del eje X, alternativamente nos ofrecían como medida 0. En este punto del proyecto perdimos aproximadamente una semana revisando código, desmontando el sensor y realizando pruebas. Finalmente, descartados los posibles problemas físicos y los errores de código, nos pusimos en contacto con el fabricante del sensor. El problema estaba en el firmware de la NXT que, para poder utilizar este sensor correctamente, debía estar actualizado a la versión 1.03 o posteriores. Con el firmware ya actualizado, el programa funcionaba correctamente.

Una de las posibles aplicaciones de los datos obtenidos del acelerómetro sería la realización de un nivel electrónico.

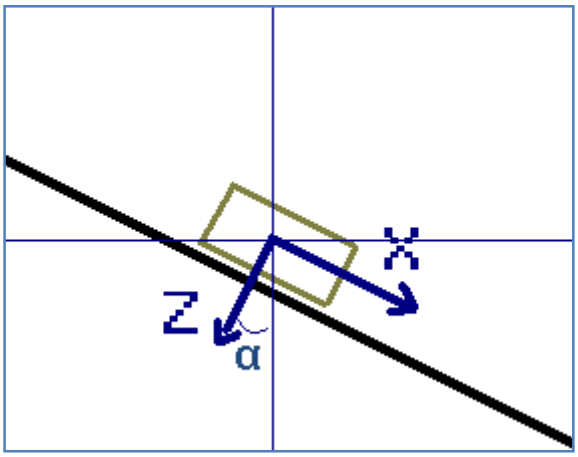

Fig 2.21 Esquema de los ejes X y Z del sensor acelerómetro sobre una superficie inclinada.

Si colocamos el montaje de la **Fig 2.19** en reposo sobre una superficie plana horizontal, deberíamos observar una medida de 200 unidades negativas (g) en el eje Z:

$$z = -200 \cdot \cos \propto \underset{\propto=0}{\longrightarrow} z = -200 \qquad \textbf{(2.1)}$$

Dónde Z sería la medida del sensor en ese eje y α el ángulo de inclinación. Se comprueba que, para una α igual a 0, es decir, con la NXT colocada sobre un plano paralelo al suelo, obtenemos un valor de Z de -200 unidades.

Así pues, en caso de colocar el montaje sobre un plano inclinado (**Fig 2.21**), podríamos calcular el ángulo de la inclinación con la siguiente fórmula:

$$\propto = \cos^{-1}\frac{z}{-200} \qquad \textbf{(2.2)}$$

Utilizando esta fórmula, creamos un programa con Labview que permite ver en pantalla los grados de inclinación en cada eje y, finalmente, visualizar la medida de los ejes X y Y en una grafica xy (**Fig 2.22**).

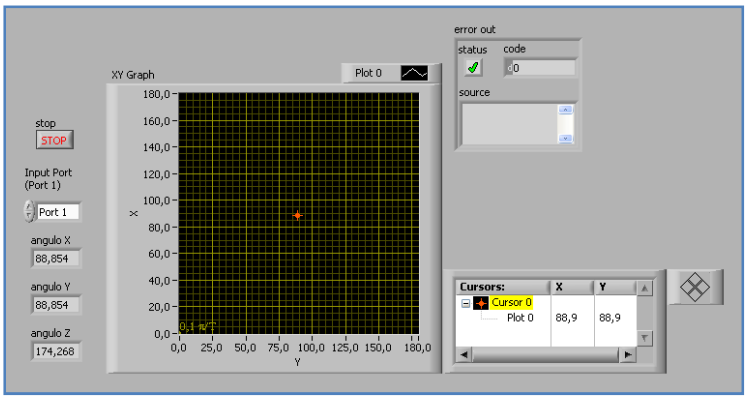

Fig 2.22 Interfaz de usuario del programa Nivel.

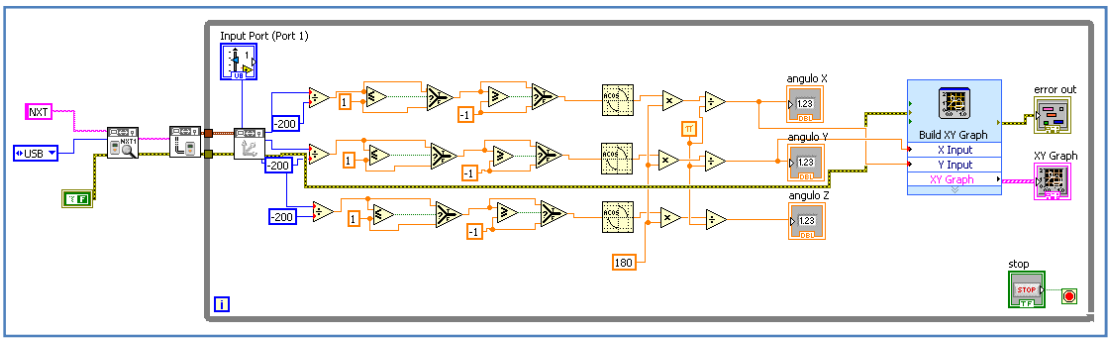

Fig 2.23 Diagrama de bloques del programa Nivel.

Inicialmente, dividimos la medida de cada eje por -200 y calculamos el arcocoseno del resultado (que había sido limitado anteriormente a un valor entre -1 y 1 para corregir medidas mayores a 200 o menores a -.200). A continuación, convertimos el resultado de radianes a grados y lo mostramos en pantalla.

Con los resultados obtenidos en esta aplicación podríamos ser capaces de nivelar una cámara de video con dos motores, uno en el plano horizontal y otro en el vertical, para que se mantuviese siempre en una posición neutra.

Por último, vamos a hablar del sensor brújula. Este sensor proporciona el sentido de la dirección a nuestro robot.

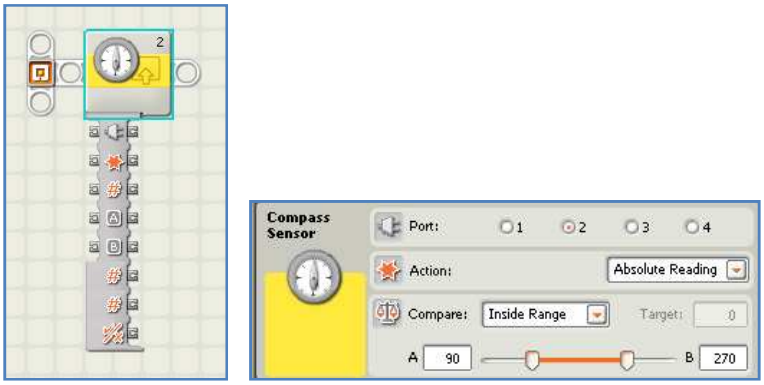

Fig 2.24 Bloque del sensor brújula para NXT-G.

El bloque del sensor brújula (**Fig 2.24**) nos permite escoger el puerto al que vamos a conectar el sensor, la acción que realiza el sensor (lectura absoluta, lectura relativa o calibración) la medida respecto a la cual queremos comparar y el rango de comparación.

Si seleccionamos la acción lectura absoluta (Absolute Reading), el sensor proporciona al robot la habilidad de conocer la dirección que está apuntando el sensor mediante un valor numérico de 0 a 359. La medida 0º indica el norte magnético, 90º indican el este, 180º indican el sur y 270º, el oeste. Podemos utilizar la sección de comparación para obtener un valor lógico (verdadero o falso).

Si seleccionamos la acción lectura relativa (Relative Reading), se nos permite fijar una dirección como objetivo (target) y, en ese caso, la medida que obtenemos del sensor varía de -180º a 180º con respecto a esa dirección.

La última opción, Calibración, no es necesaria realizarla cada vez que vayamos a utilizar el sensor brújula, ya que este sensor es extremadamente fiable y no necesita ser calibrado. Sin embargo, es posible que el sensor funcione mal si se coloca demasiado cerca de otros componentes eléctricos o motores. En ese caso, es aconsejable utilizar el bloque en modo calibración para corregir el mal comportamiento del sensor al inicio de un programa.

En primer lugar, vamos a crear un programa muy sencillo que nos permita ver en la pantalla de la NXT la medida del sensor brújula (**Fig 2.25**).

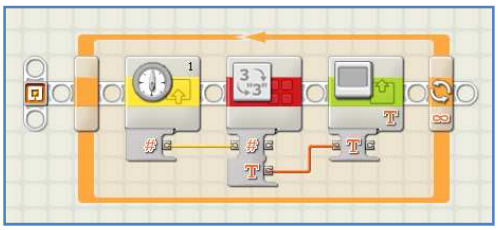

Fig 2.25 Diagrama de bloques del programa Brújula.

Si colocamos el sensor sobre una superficie plana y lo giramos, observamos en pantalla como varia el valor numérico de 0 a 359º, ya que en este caso estamos utilizando el sensor en modo lectura absoluta.

A continuación vamos a realizar un pequeño programa que conduzca a nuestro robot hacia el norte. En la **Fig 2.26** se observa el montaje que vamos a utilizar para el programa.

Fig 2.26 Montaje del programa Norte.

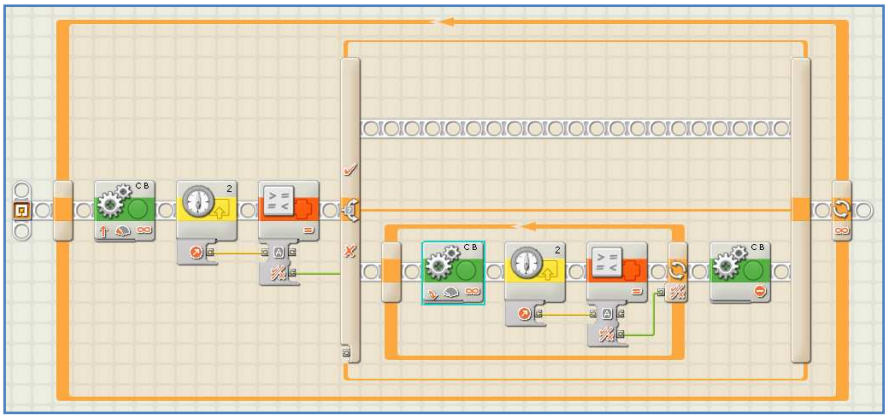

Fig 2.27 Diagrama de bloques del programa Norte.

Tal como se observa en la **Fig 2.27**, el robot se mantiene en dirección fija siempre que el sensor brújula ofrece una medida igual a 0º (norte). Cuando no es así, el robot gira sobre si mismo hasta que recupera la dirección correcta.

Conocimientos adquiridos

Esta actividad permite:

- Aprender cuales son los datos que nos ofrece un giróscopo y un acelerómetro, y su utilidad.
- Aprender a tratar y transformar los datos recibidos por estos sensores para obtener otros datos que nos fueran de utilidad.
- Aprender a utilizar Labview para programar la NXT.

2.3. Proyectos avanzados

Entendemos por proyectos avanzados aquellos en los que utilizaremos alguno o algunos de los sensores avanzados del apartado anterior y los datos que nos aportan para crear un nuevo robot, distinto al resto de aplicaciones que podamos encontrar en internet o en algunos libros guía.

Como ejemplo, nos hemos propuesto realizar un robot-vehículo controlado por un mando semejante a una wii, es decir, controlado por nuestro movimiento. Para ello vamos a utilizar lo siguiente:

- 2 NXT
- 1 sensor Acelerómetro
- 2 motores

Una de las NXT junto con los dos motores formará el montaje de NXT-vehículo, mientras que la otra NXT junto al sensor Acelerómetro formará el montaje de NXT-mando (**Fig 2.28**).

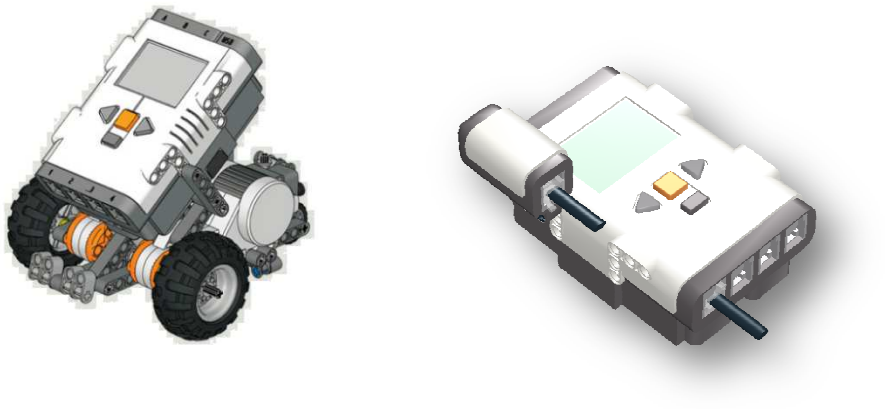

Fig 2.28 Montaje de la NXT-vehículo (izquierda) y de la NXT-mando (derecha).

Después de realizar los montajes, el siguiente paso será la programación de las dos NXT. El primer paso para realizar el programa mando será definir los límites de medida del acelerómetro, que serán los cambios de dirección de nuestro vehículo. Para realizar estas medidas podemos utilizar el programa Accel Test, que mostraba en pantalla el valor de la medida del sensor Acelerómetro en los tres ejes (**Tabla 2.1**).

Tabla 2.1 Límites de medida y códigos numéricos.

Dirección	Medida Acelerómetro	Código numérico
Adelante	Eje Z > 180	1
Atrás	Eje Z < -180	2
Derecha	Eje Y > 180	3
Izquierda	Eje Y < -180	4

Para facilitar la comunicación entre las dos NXT, que se realizará mediante mensajes bluetooth, hemos adjudicado un código numérico para cada una de las direcciones.

Por tanto, cuando el acelerómetro conectado a la NXT-mando detecte una medida superior a 180 o inferior a -180 en uno de los ejes Z o Y, enviará un mensaje con el código numérico correspondiente vía bluetooth. El NXT-

vehículo se mantendrá siempre a la espera de un nuevo mensaje bluetooth para activarse o cambiar de dirección. Para parar y apagar la NXT-vehículo pulsaremos el botón naranja de la NXT-mando, y enviaremos el código “0”.

El programa NXT-mando (**Fig 2.29**) consta de un bucle en que leemos la medida de los ejes Y y Z del bloque acelerómetro, para después comparar si es mayor que 180 o menor que -180 y, en caso de ser cierta alguna de las comparaciones, enviar por bluetooth el código correspondiente a la acción que debe realizar el vehículo. Por último, también compararemos si el botón naranja de la NXT está presionado y, en caso de ser cierto, enviaremos el código “0” o de parada por bluetooth y saldremos del bucle.

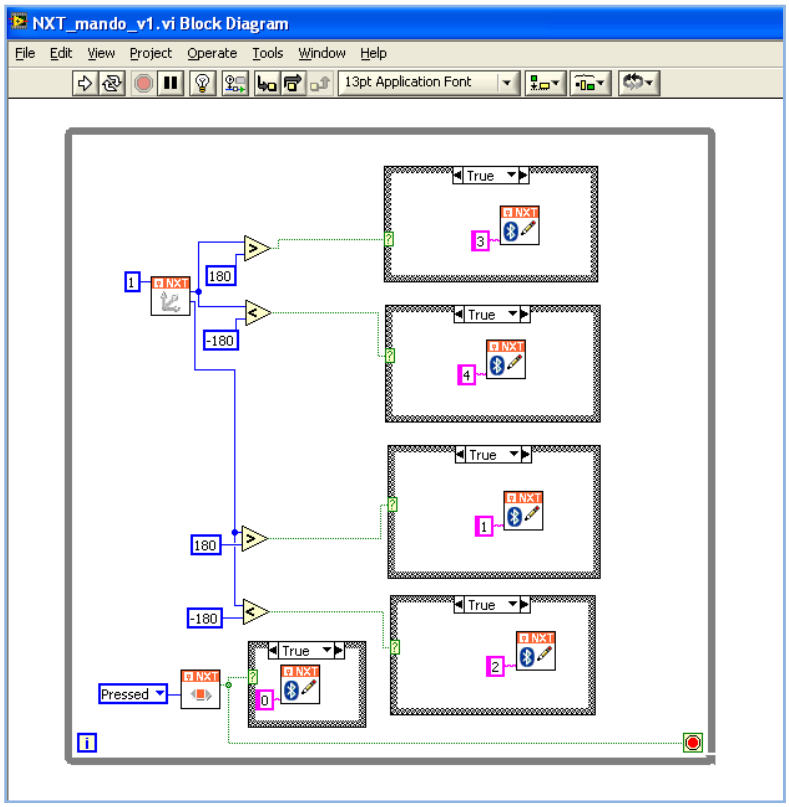

Fig 2.29 Diagrama de bloques del programa NXT_mando_v1.vi.

El programa NXT-vehículo consta de un bucle en que comprobamos si ha llegado algún mensaje nuevo al buzón del bloque de recepción de mensajes bluetooth. En caso de ser cierto, convertimos el texto recibido en un número y comparamos si este es igual a alguno de los posibles códigos numéricos y actuamos en consecuencia, es decir, si es igual a 1, hacemos que los motores avancen hacia delante, si es igual a 2 hacemos que se mueva marcha atrás, etc. Por último, si es igual a 0, paramos los motores y salimos del bucle.

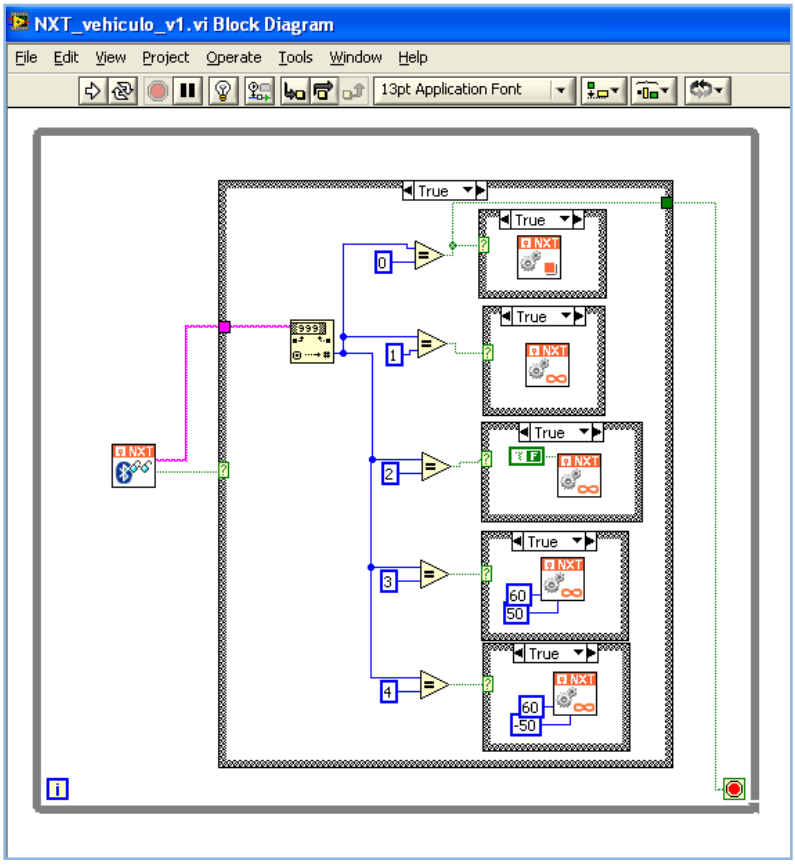

Fig 2.30 Diagrama de bloques del programa NXT_vehículo_v1.vi.

El resultado de la prueba de este primer prototipo ha sido correcto, pero se han observado algunas posibles mejoras que serian interesantes para el manejo de la NXT-vehículo:

- Definir una posición de reposo en que los motores de la NXT-vehículo se mantuvieran parados, sin necesidad de parar el programa.
- Establecer un botón que, al permanecer presionado, mantiene parados los motores de la NXT-vehículo en cualquier posición.
- Añadir un símil del conocido nunchuk de una wii a la NXT-mando. El nunchuk estará formado por el sensor de tacto y por el sensor giróscopo, y lo utilizaremos para mover un tercer motor que añadiremos a la NXT-vehículo y que hará de brazo.

Por tanto, para llevar a cabo un segundo prototipo de este proyecto con las mejoras anteriores, será necesario modificar los dos montajes (**Fig 2.31**).

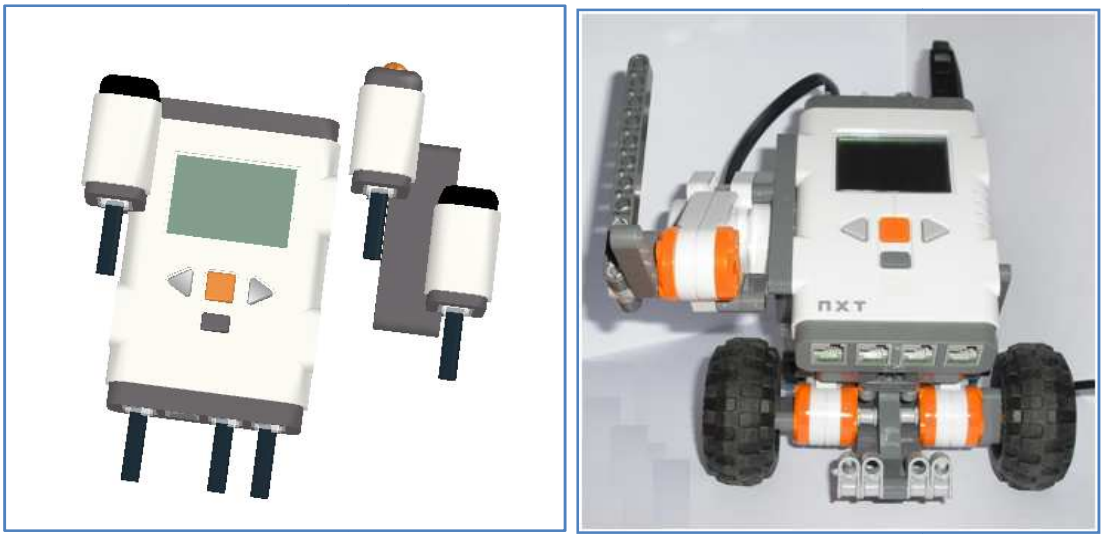

Fig 2.31 Montaje de la nueva NXT-mando (izquierda) y de la nueva NXT-vehículo (derecha).

Después de realizar los nuevos montajes, el siguiente paso será modificar la programación de las dos NXT. Establecemos como posición de reposo una medida del eje X del bloque acelerómetro mayor a 190 y le asignamos el código numérico -1.

Para mover el brazo de la NXT-vehículo, pulsaremos el sensor de tacto, moveremos el giróscopo y volveremos a presionar el sensor de tacto. La suma total de los grados girados será enviada para mover el tercer motor o brazo.

El problema que surge aquí es el siguiente: ¿Cómo enviamos la medida de grados girados asegurándonos de que esta no será igual a uno de nuestros códigos numéricos? Será necesario procesar los datos de la medida del giróscopo para no confundirlos con el resto de códigos. Puesto que nuestro código numérico está formado por valores que van del -1 al 4, multiplicaremos la media de grados girados por 10 y así aseguraremos que cada vez que recibamos un dato mayor a 10 o menor a -10 (puesto que los grados girados pueden ser negativos) esos datos recibidos serán grados que hay que girar en el tercer motor de la NXT-vehículo.

El programa NXT-mando (**Fig 2.32**) constará del mismo planteamiento que el prototipo 1 pero con los siguientes añadidos:

- Compararemos también si la medida del eje X en el bloque acelerómetro es mayor a 190 y, en caso de ser cierto, enviaremos por bluetooth el código numérico -1.
- De igual forma, si pulsamos el botón izquierdo de la NXT-mando, enviaremos por bluetooth el código numérico -1.

- Si pulsamos el sensor de tacto, entraremos en un bucle en que calcularemos la suma total de los grados girados hasta pulsar de nuevo el sensor (programa *cuenta_grados_NXT.vi* creado anteriormente). El resultado total se multiplicará por 10 y se enviará por bluetooth.
- La última funcionalidad que se ha añadido al programa es la impresión en la pantalla de la NXT-mando de la orden que se está enviando en todo momento a la NXT-vehículo (Forward, Backward, Right, Left, Stop and Gyro).

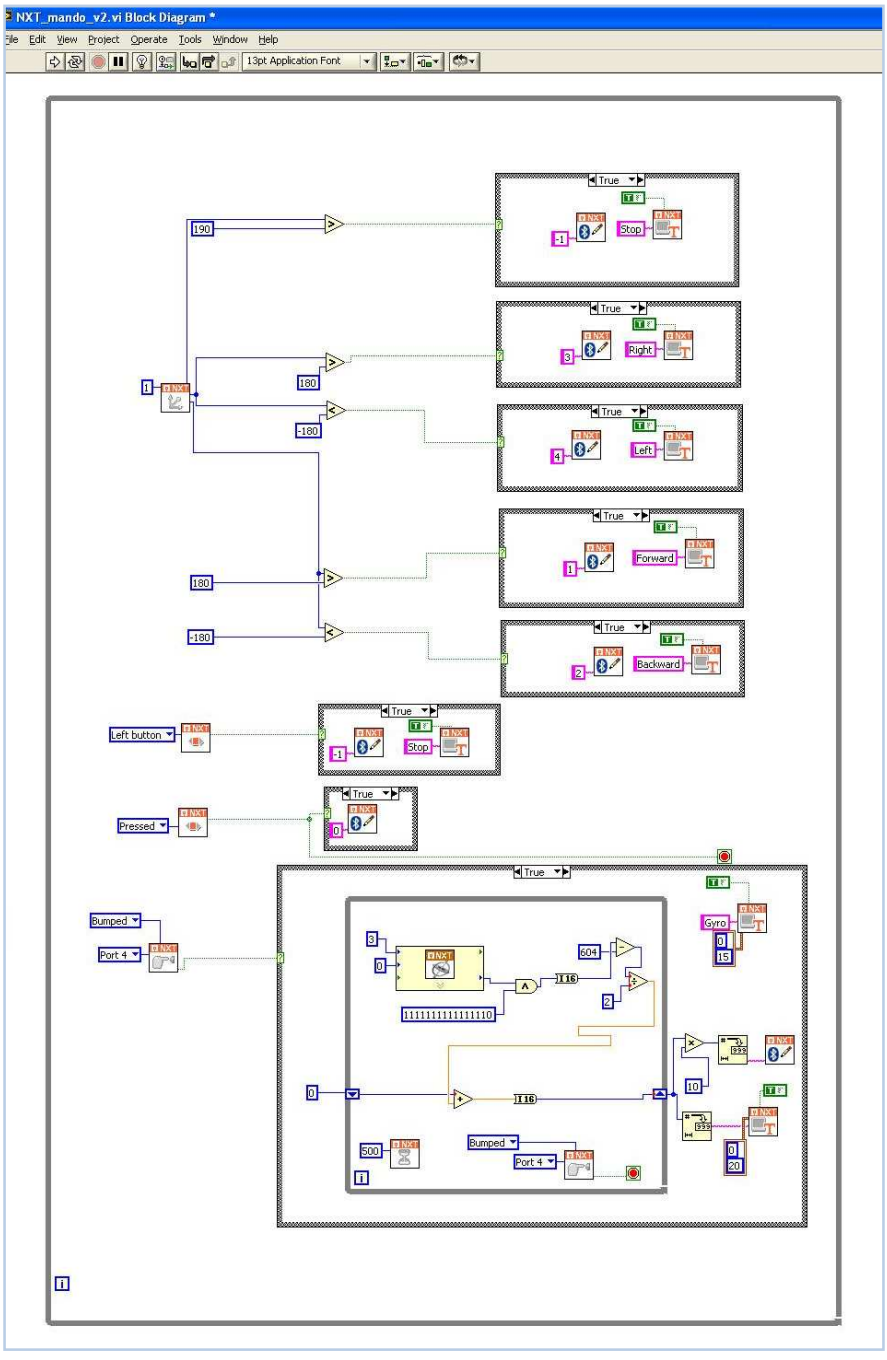

Fig 2.32 Diagrama de bloques del programa NXT_mando_v2.vi.

El programa NXT-vehículo (**Fig 2.33**) constará del mismo planteamiento que el prototipo 1 pero con los siguientes añadidos:

- Si el código numérico recibido es igual a -1, pararemos los motores.
- Si el código numérico recibido es mayor a 10 o menor a -10, lo dividiremos por 10 y el resultado será el número de grados que habrá que mover el tercer motor (si son positivos hacia delante y en caso contrario hacia atrás).

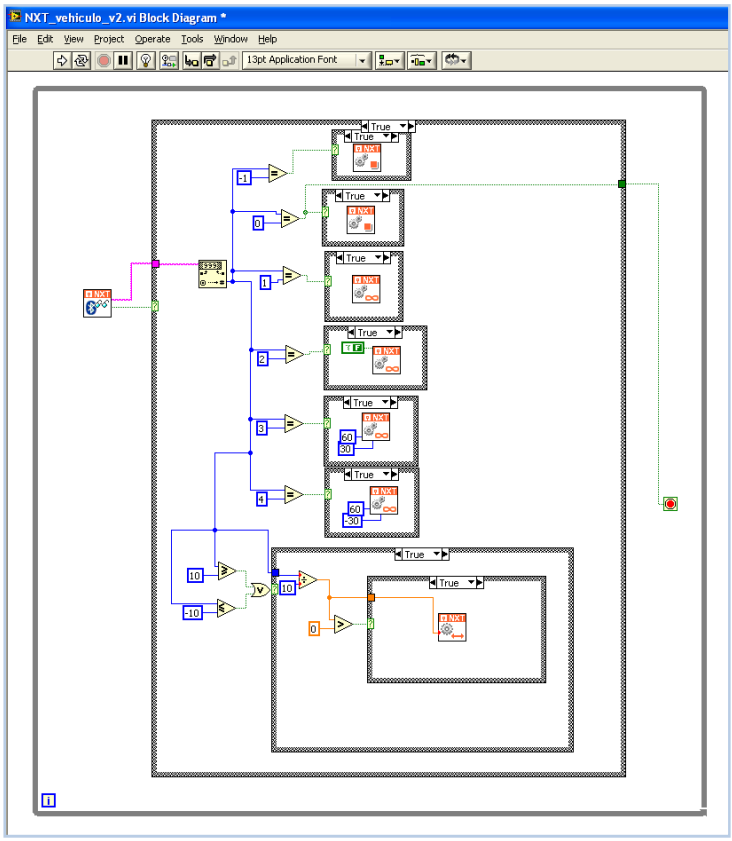

Fig 2.33 Diagrama de bloques del programa NXT_ vehículo_v2.vi.

El resultado de la prueba de este segundo prototipo ha sido satisfactorio, puesto que hemos conseguido realizar todas las propuestas, pero se han observado algunas posibles mejoras nuevas que serian interesantes para el manejo de la NXT-vehículo:

- Que el tercer motor, o brazo, de la NXT-vehículo, reproduzca exactamente nuestros movimientos con el nunchuk y no solo la suma final de grados girados.
- Que la velocidad de la NXT-vehículo varíe gradualmente en función de la inclinación de la NXT-mando.

En este caso, para llevar a cabo un tercer prototipo de este proyecto con las mejoras anteriores, no será necesario modificar ninguno de los montajes.

El primer paso será modificar la programación de las dos NXT. Puesto que queremos que la velocidad aumente a medida que la inclinación es mayor, necesitaremos determinar un método para tratar los datos de inclinación y convertirlos a velocidad. Anteriormente, con una inclinación mayor a 180 (o menor a -180) en los ejes Y y Z, detectábamos un cambio de dirección. Puesto que la medida de la inclinación llega hasta un máximo de 200 (o mínimo de -200) que es el valor de la gravedad, restaremos 100 a la medida de la inclinación para obtener el valor de potencia del motor (que va de 0 a 100). En este caso, solo tendremos en cuenta las medida de inclinación por encima de 175 (o por debajo de -175), y, por tanto, la potencia del motor variará a medida que aumentemos la inclinación de 75 a 100. La posición de reposo continuará siendo de 190 en el eje X.

Para poder enviar la medida exacta en el programa NXT-mando, enviaremos primero el código numérico y después la medida de la inclinación – 100 (**Fig 2.34**).

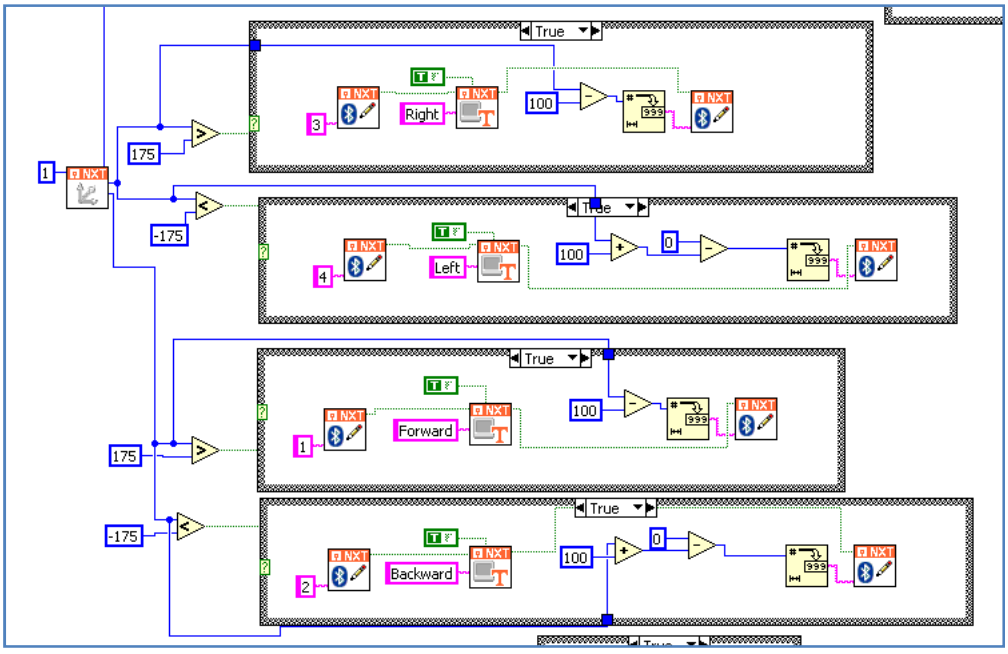

Fig 2.34 Diagrama de bloques del programa NXT_mando_v3.vi, parte 1 de 2.

En el programa NXT-vehículo (**Fig 2.35**), primero recibiremos el código numérico para descifrar la dirección, y después esperaremos la recepción de la potencia que se debe aplicar a los motores.

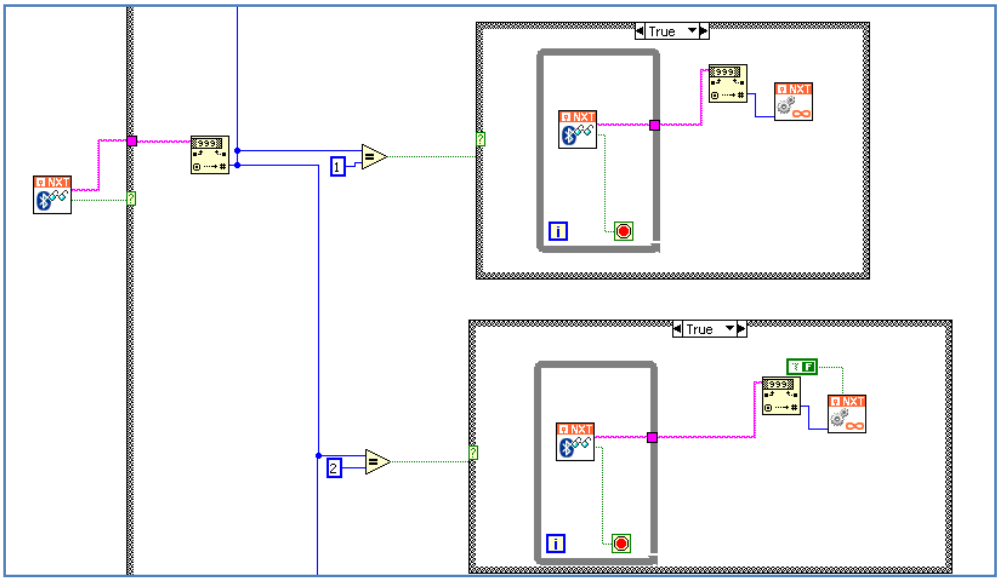

Fig 2.35 Diagrama de bloques del programa NXT_vehículo_v3.vi, parte 1 de 2.

Para conseguir que el tercer motor de la NXT-vehículo se mueva a la vez que movemos el nunchuk, necesitaremos un nuevo código numérico que enviaremos primeramente en el programa NXT-mando (**Fig 2.36**). De esta forma, el programa NXT-vehículo sabrá que los datos que va a recibir a continuación son grados girados con el giróscopo. Después de enviar el código, que en este caso será “5”, enviamos cada medio segundo los grados que giramos, hasta que volvemos a apretar el botón del sensor de tacto y volvemos a enviar el código. Antes de enviar los grados girados, los multiplicamos por 10 para, en recepción, poder distinguirlos del código “5” que indica que ya hemos acabado de utilizar el nunchuk.

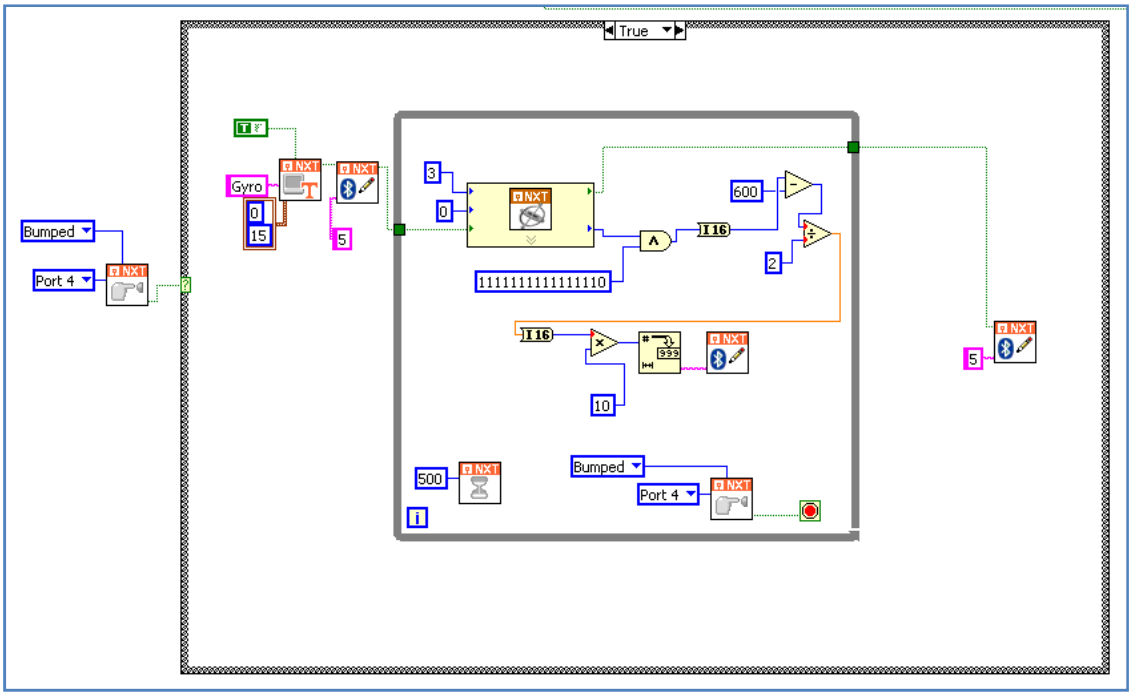

Fig 2.36 Diagrama de bloques del programa NXT_mando_v3.vi, parte 2 de 2.

En el programa NXT-vehículo (**Fig 2.37**), cuando se recibe el código 5, entramos en un bucle y esperamos a recibir el próximo mensaje bluetooth que, si no es 5 (no se ha vuelto a pulsar el botón), se divide por 10 y se sigue el mismo proceso que anteriormente. En caso de ser 5 alguno de los mensajes recibidos, se sale del bucle.

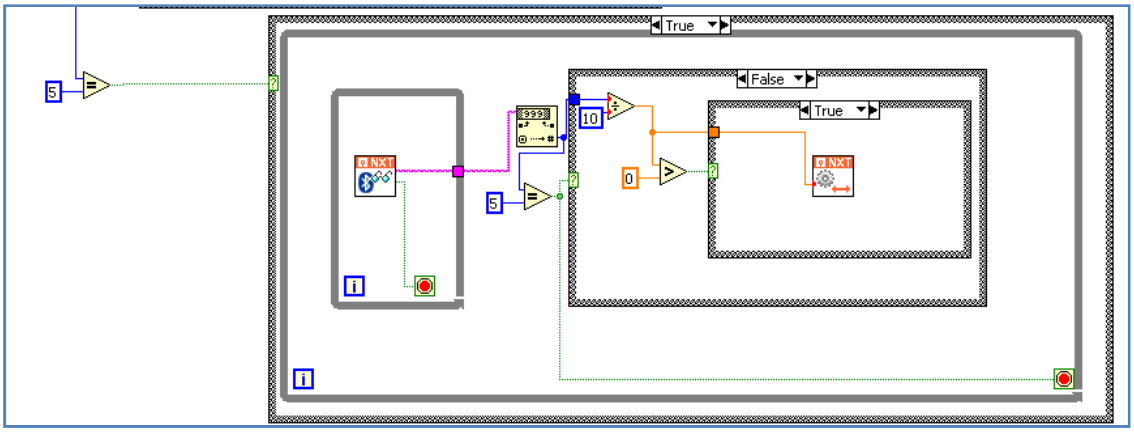

Fig 2.37 Diagrama de bloques del programa NXT_vehículo_v3.vi, parte 2 de 2.

Durante la prueba de este tercer prototipo se ha observado un movimiento constante del tercer motor de la NXT-vehículo cuando el nunchuk estaba

accionado (el botón había sido pulsado) pero en reposo. Este fallo se debe a que estábamos aplicando de nuevo, como ya había ocurrido en el apartado **2.2 Bloques para sensores avanzados**, un offset erróneo a la medida del giróscopo. Para que esto no vuelva a suceder, lo ideal sería realizar una calibración inicial en el programa NXT-mando (**Fig 2.38**).

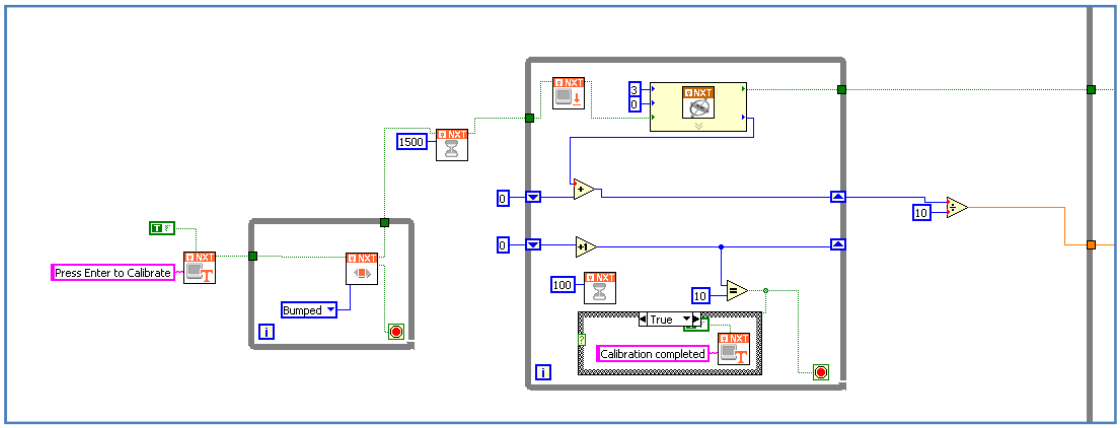

Fig 2.38 Mejora introducida en el programa NXT_mando_v3.vi.

Al empezar el programa, aparece en pantalla el mensaje “Press Enter to Calibrate”. Esto debería darnos tiempo para colocar el nunchuk junto a la NXT y dejar el conjunto en reposo. Al pulsar el botón naranja de la NXT-mando, se realizan 10 medidas del valor del giróscopo que se van sumando y, al acabar la décima, aparece en pantalla “Calibration completed”, es decir, ya podemos mover el nunchuk. Se divide la suma por 10 y el resultado de esta media lo utilizaremos como offset del giróscopo durante el resto del programa.

Conocimientos adquiridos

Esta actividad permite:

- Aprender a conectar dos NXT mediante bluetooth.
- Aprender a procesar los datos antes de enviarlos para ser capaces de distinguirlos, mediante un código establecido por nosotros.
- Aprender a utilizar los conocimientos sobre sensores avanzados adquiridos en el apartado anterior para una aplicación final.

2.4. Diseño y montajes hardware

Para realizar nuestros propios sensores y actuadores para LEGO Mindstorms, es necesario dar un paso más, en cuanto al conocimiento de nuestra herramienta. Hasta el momento no era necesario conocer como está diseñado cada sensor o motor, o siquiera como se comunican estos con el ladrillo y viceversa, ya que los bloques del NXT-G, Labview, o las librerías creadas para NBC y NXC nos eran suficientes para utilizar-los. A continuación, conoceremos como conectar a la NXT sensores o actuadores construidos por nosotros mismos.

Si nos fijamos en las conexiones al final de uno de los cables de conexión de la NXT (**Fig 2.39**), podemos ver seis pines y cables que crean la interfaz. Los cables siguen un código de color: blanco, negro, rojo, verde, amarillo y azul.

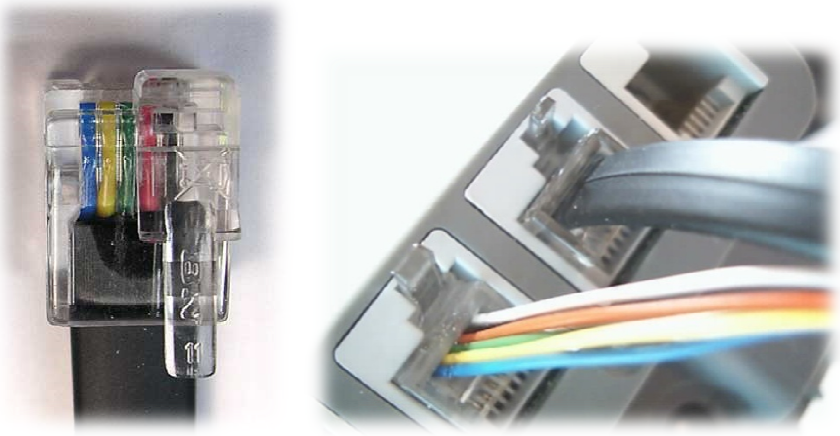

Fig 2.39 Imágenes de los cables de conexión de la NXT.

La función de estos cables es diferente si se usa como entrada de sensor o como salida de motor. También depende del tipo de sensor que se le conecta.
La **Tabla 2.2** resume los colores y los nombres de los pines para los **puertos de entrada de sensores.**

Tabla 2.2 Puertos de entrada de sensores.

Puertos de entrada de sensores		
Número de Pin	Color	Nombre
1	Blanco	AN
2	Negro	Tierra
3	Rojo	Tierra
4	Verde	4,3V Potencia
5	Amarillo	DIGI0
6	Azul	DIGI1

- **1. AN:** Este pin se puede usar para dos propósitos: como entrada analógica o como una fuente de energía de 9V usada para la compatibilidad con los antiguos sensores del RCX.
  Cuando utilizamos el pin como entrada analógica, la señal está conectada a un conversor analógico digital de 10 bits (**Fig 2.40**). La señal de entrada debe estar en el rango de 0 a 5V y se convertirá en un valor digital de entre 0 y 1023. El valor es muestreado cada 3 ms. El pin está permanentemente conectado a 5V de tensión a través de una resistencia de arranque de 10kΩ.

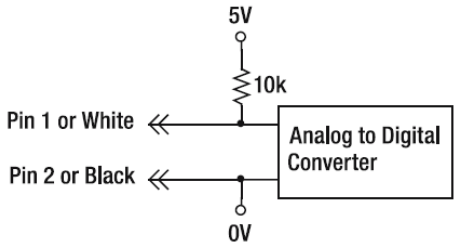

Fig 2.40 Entrada de sensor de NXT.

  Si utilizamos el pin como una fuente de energía de 9V (voltaje de las baterías), podemos alimentar los antiguos sensores del RCX. La NXT alimenta el sensor 3 ms y después lee el valor durante 0,1 ms. El sensor necesita un condensador para almacenar la potencia durante el intervalo de lectura.
- **2 y 3. Tierra:** Estos dos pines que están conectados juntos en la NXT y en los sensores de LEGO son los pines de tierra. Podemos utilizar uno de ellos, o los dos a la vez.

- **4. 4.3V Potencia:** Esta es la fuente de energía principal para todos los sensores y actuadores de la NXT. Esta fuente de energía tiene una corriente límite de 180mA para los siete puertos de la NXT. Cada puerto puede utilizar 25mA de media, pero también es posible que un puerto consuma más corriente si otro consume menos.
- **5 y 6. DIGI0 y DIGI1:** Estos pines son señales de 3.3V conectadas directamente al microprocesador de la NXT. Se usan principalmente para las comunicaciones I^2C. Para limitar la corriente, este puerto tiene una resistencia de 4.7kΩ conectada en serie.

La **Tabla 2.3** resume los colores y los nombres de los pines para los **puertos de salida de motores.**

Tabla 2.3 Puertos de salida de motores.

Puertos de salida de motores		
Número de Pin	**Color**	**Nombre**
1	Blanco	M1
2	Negro	M2
3	Rojo	Tierra
4	Verde	4,3V Potencia
5	Amarillo	TACH00
6	Azul	TACH01

- **1 y 2. M1 y M2:** Estos pines alimentan a los motores con una tensión máxima de 9V (la de las baterías). El motor está controlado por un circuito llamado Puente-H (**Fig 2.41**). Este puente está formado por 4 transistores (Q1, Q2, Q3 y Q4). El circuito de control está diseñado de tal forma que los transistores 1 y 2 en un lado y 3 y 4 en otro lado no están conduciendo simultáneamente. La **Fig 2.41** muestra el estado de los transistores cuando el motor va hacia delante, hacia atrás y cuando está parado.

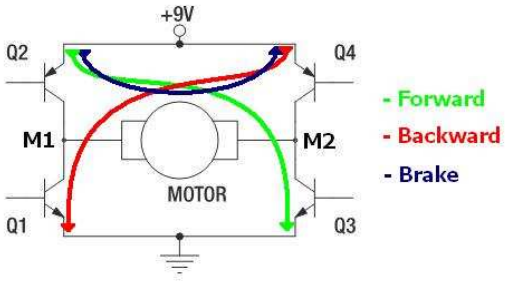

Fig 2.41 Puente H y estados del motor.

La velocidad del motor se controla por una modulación PWM (Pulse Width Modulation), como se muestra en la **Fig 2.42**.

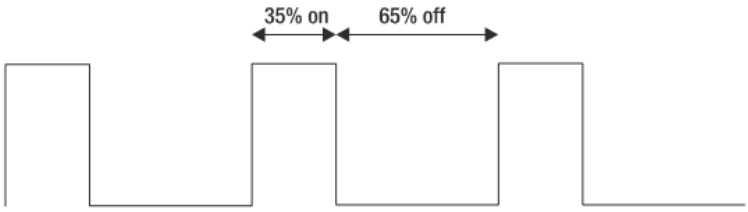

Fig 2.42 Pulse Width Modulation para el motor a potencia del 35%.

La alimentación del motor cambia rápidamente de on a off durante un intervalo de tiempo (ciclo de 128 µs, 7800 Hz). La velocidad del motor depende de la media del voltaje aplicado sobre él. En la NXT, la relación entre la velocidad y el voltaje aplicado es lineal.

- **3. Tierra:** Este es el pin de tierra. A diferencia de los pines de los sensores, este no está conectado con el pin número 2. Si conectásemos un sensor en un puerto de motor, el driver estaría siendo parcialmente cortocircuitado. Aunque el driver está bien protegido, es muy recomendable no hacerlo.
- **4. 4.3V Potencia:** Como se ha dicho anteriormente, esta es la fuente de energía principal para todos los sensores y actuadores de la NXT. Esta fuente de energía tiene una corriente límite de 180mA para los siete puertos de la NXT. Cada puerto puede utilizar 25mA de media, pero también es posible que un puerto consuma más corriente si otro consume menos.
- **5 y 6. TACH00 y TACH01:** Estos dos pines se utilizan para el codificador óptico incluido en los motores de la NXT. El codificador genera señales en cuadratura y esto permite a la NXT determinar la dirección y la velocidad del motor. Las dos señales son pulsos

rectangulares desplazados, y el desplazamiento representa un cuarto de fase. La **Fig 2.43** muestra la señal para un motor rotando hacia adelante (desfase positivo).

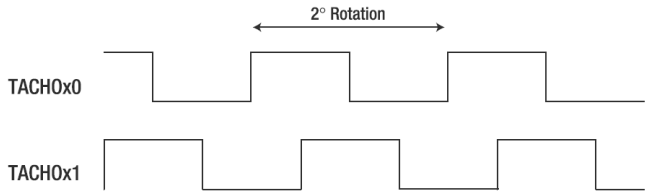

Fig 2.43 Señales en cuadratura para el motor funcionando hacia adelante.

La **Fig 2.44** muestra la señal para un motor rotando hacia atrás (desfase negativo).

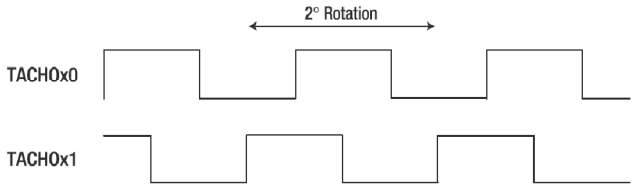

Fig 2.44 Señales en cuadratura para el motor funcionando hacia atras.

La frecuencia de las señales nos da la velocidad de rotación del motor. Medio ciclo de la señal corresponde a un grado de rotación del motor.

A continuación, vamos a aplicar lo que hemos aprendido hasta el momento. Para ello vamos a construir nuestro propio sensor de luz a partir de un LDR (Light Dependent Resistor). Un LDR no es más que una fotorresistencia, es decir, una resistencia cuyo valor varía en función de la intensidad de luz que recibe. Los valores que puede tomar una LDR en general oscilan entre unos 50Ω y 1kΩ cuando están iluminadas y valores comprendidos entre 50kΩ y varios megohmios cuando está a oscuras.

En la **Fig 2.45** se observa el circuito que tendríamos montado si conectamos la LDR entre los pines 1 (blanco, AN) y 2 (negro, Tierra) de una entrada de sensor.

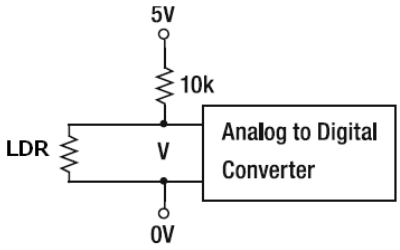

Fig 2.45 Esquema de la conexión del LDR al conversor A/D.

Se trata de un divisor de tensión, y la tensión que tendríamos en los bornes de la LDR se calcula con la siguiente fórmula:

$$V_{LDR} = \frac{R_{LDR}}{10000+R_{LDR}} \cdot 5 \qquad \textbf{(2.3)}$$

Sin embargo, el conversor analógico digital convierte esta tensión en un valor (o Raw) de 0 a 1023 (Obtendremos el valor “Raw” del bloque del sensor de tacto):

$$Raw = \frac{R_{LDR}}{10000+R_{LDR}} \cdot 1023 \qquad \textbf{(2.4)}$$

Por tanto, la R_{LDR} se calcula con la siguiente fórmula:

$$R_{LDR} = \frac{10000 \cdot Raw}{1023-Raw} \qquad \textbf{(2.5)}$$

En primer lugar, vamos a conectar la LDR a la NXT con la ayuda de una protoboard. Cortamos uno de los cables conectores de la NXT y retiramos un trozo del plástico negro, dejando al descubierto los 6 cables de colores. Finalmente, conectamos la LDR entre el cable blanco y el negro.

A continuación, vamos a crear un pequeño actuador que podremos utilizar con nuestro sensor. Vamos a conectar un LED a una salida de motor de la NXT para controlar la cantidad de luz que emite a través de la modulación PWM. Es decir, a través del bloque motor y de su entrada “Power” podremos controlar la media de tensión con que alimentaremos al LED.

Con la ayuda de la protoboard, vamos a conectar el LED entre los pines 1 (blanco, M1) y 2 (negro, M2), de la misma forma que en el LDR solo que en este caso se trata de pines de salida de motor y no de entrada de sensor (**Fig 2.46**).

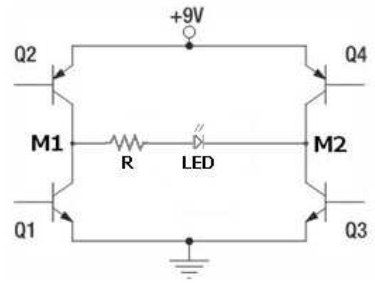

Fig 2.46 Esquema de la conexión del LED al puente H.

La tensión màxima que recibirà este LED será de 9V, así que deberemos colocar una resistencia en serie para límitar la corriente. Calculamos el valor de la resistencia para una corriente màxima de 15mA:

$$R = \frac{9V}{15 \cdot 10^{-3} A} = 600\Omega \qquad \textbf{(2.6)}$$

Con nuestro nuevo sensor y actuador, hemos realizado un proyecto llamado Ahorra luz. El robot mantiene siempre una determinada cantidad de luz sobre un punto determinado, aportando luz con nuestro actuador si la luz recibida está por debajo del mínimo que se requiere o disminuyendo la potencia del LED a medida que aumenta la cantidad de luz natural. La documentación completa de este proyecto se puede encontrar en el **Annexo I**.

Conocimientos adquiridos

Esta actividad permite:

- Aprender como se comunican los sensores y los actuadores con la NXT.
- Aprender el montaje interno de los sensores y el montaje interno de la NXT (conversor A/D, alimentación, divisor de tensión).
- Aprender a crear nuestro propio sensor y nuestro propio actuador, así como a programarlos a través de los bloques de otros sensores.

CONCLUSIÓN

Cuando inicié este trabajo, con el pack de LEGO Mindstorms NXT en mis manos, tenía claro cuál era el principal objetivo: investigar el uso del pack de LEGO como herramienta de aprendizaje en la ingeniería. Sin embargo, no sabía por dónde debía empezar ni cuáles eran los pasos a seguir para llegar a conseguir este objetivo. Tampoco sabía entonces que dificultades o que sorpresas iban a aparecer en mi camino. Así pues, mi trabajo, como el de todo ingeniero, consistió en investigar, desarrollar, diseñar y crear, aunque en un principio no tuviese la certeza de que mis pasos eran los correctos.

A lo largo de la carrera, los profesores nos proponen una serie de tareas y trabajos. Para la realización de ellos, acostumbran a darnos una serie de pautas, o incluso pistas, que seguir. Sin embargo, la tarea más difícil siempre es organizar, planificar y decidir las acciones a realizar en un proyecto, y eso es algo que he aprendido en este trabajo.

Debo aclarar que, al iniciarlo, no tenía ningún tipo de experiencia con robots ni había jugado nunca con uno de los packs de LEGO. Así pues, mis primeros pasos con el robot fueron los de un principiante: observar, aprender y reproducir diseños existentes. Después de haber comprendido el modo de programación del robot con el programa NXT-G proporcionado por el fabricante, y haber probado diferentes montajes, realicé mis primeros robots sencillos. Esta parte del trabajo fue la más sencilla, ya que hay mucha información acerca de esta herramienta, y el lenguaje gráfico de NXT-G simplifica mucho la programación.

El siguiente paso llegó con el giróscopo y con el acelerómetro de Hitechnic. Puesto que desconocía la idea de giróscopo y de acelerómetro, lo primero que hice fue informarme acerca de la utilidad de estos sensores, los datos que me ofrecían y algunos ejemplos reales en que se utilizaban.

Más adelante, surgió la necesidad de cambiar el software NXT-G por una herramienta de programación más avanzada. El software NXT-G era limitado en cuanto a interfaz de usuario y poco manejable en la creación de programas de tamaño considerable. Es por eso que se decidió cambiar a Labview, el software en que está basado NXT-G, una herramienta más potente que utiliza igualmente lenguaje gráfico y que se usa ampliamente en ámbitos profesionales y educacionales. Aprender a manejar Labview no es tan trivial como aprender a manejar NXT-G, pero mereció la pena el esfuerzo, puesto que Labview resolvió la mayoría de los problemas que causaba NXT-G: falta de funciones, imposibilidad de crear vectores, difícil conexión con la NXT por

bluetooth, etc. A partir de aquí, parecía importante crear una aplicación que englobase todo lo aprendido hasta la fecha y surgió la idea de emular un mando de wii, ya que disponíamos de acelerómetro y giróscopo. Puesto que la NXT iba a simular el mando, necesitábamos una segunda NXT que recibiese las órdenes por la conexión bluetooth. A medida que el proyecto crecía y se añadían funcionalidades, aumentaba la dificultad de transmitir los datos de una NXT a la otra y seguir siendo capaces de distinguirlos. Con este proyecto entendí la dificultad que tiene la transmisión de datos y la importancia del cifrado de datos.

Finalmente, profundicé en el montaje interno de la NXT, de los sensores, de los conectores de los cables y comprendí la forma en que se comunicaban sensores y actuadores con la NXT. Es importante darse cuenta de todo el trabajo que se pudo realizar hasta entonces, sin necesidad de conocer este funcionamiento interno. El montaje de nuestro propio sensor y actuador y el diseño de los dos circuitos no fue difícil, ya que el tema está muy bien documentado. Sin embargo, la realización del proyecto Ahorra luz, en que manteníamos un nivel de luz sobre un punto compensándolo siempre que el nivel de luz natural no era suficiente, fue bastante complicada en cuanto al planteamiento de la programación y al ajuste del comparador de histéresis resultante.

Mi balance final sobre la experiencia adquirida con el robot de LEGO es muy positivo, puesto que es evidente que muchos de los conceptos aprendidos durante la carrera han cobrado vida con los robots y han resultado ser, en muchas ocasiones, la solución a problemas que aparecían sin haberlos previsto. Además, me ha permitido conocer un poco más de cerca la robótica, que hasta entonces desconocía. Este trabajo también me ha permitido conocer la programación en lenguaje gráfico y, sobretodo, aprender a utilizar la herramienta Labview. Por tanto, opino que sería muy positivo para los futuros estudiantes de ingenierías que se introdujese esta metodología de aprendizaje basada en robots en las aulas.

Por todo ello, entre las virtudes de este kit educativo destacaría las siguientes:
- El LEGO Mindstorms NXT es una herramienta atractiva que permite aprovechar las ventajas de un aprendizaje basado en proyectos.
- Resolver los problemas que plantea el diseño de sistemas que “han de acabar funcionando” requiere el uso conjunto de conocimientos adquiridos en distintas asignaturas.
- La resolución de los problemas que van surgiendo al resolver los proyectos a menudo requieren conocimientos de los que no se dispone. El reto de terminar un proyecto que uno mismo plantea resulta muy motivador para buscar la información necesaria y ampliar los conocimientos propios.

BIBLIOGRAFÍA

[1] Perdue, D.J., *The Unofficial LEGO Mindstorms NXT Inventor's Guide,* Megan Dunchak, United States of America (2008).

[2] Floyd Kelly, J., *Compass Sensor Experiments*, The NXT Step Storefront (2007)

[3] Gasperi, M., Hurbain, P., Hurbain, I., *Extreme NXT*, Jim Sumser, United States of America (2007).

[3] http://es.wikipedia.org/wiki/Rob%C3%B3tica_pedag%C3%B3gica (29 de setiembre de 2009)

[4] http://www.fodweb.net/robotica/roboteca/articulos/pdf/robotica_pedagogica.pdf (29 de setiembre de 2009)

[5] http://www.donosgune.net/2000/dokumen/EduRobSp.pdf (29 de setiembre de 2009)

[6] http://www.lego.com/education/school/default.asp?locale=2057&pagename=ict_home&l2id=3_2&domainredir=www.mindstormseducation.com (13 de agosto de 2009)

[7] http://mindstorms.lego.com/eng/Israel_dest/default.aspx (30 de setiembre de 2009)

[8] http://bricxcc.sourceforge.net/nbc/ (4 de mayo de 2009)

[9] http://www.hitechnic.com/ (8 de abril de 2009)

[10] http://www.botmag.com/articles/10-31-07_NXT.shtml (21 de agosto de 2009)

[11] http://forums.nxtasy.org/ (30 de junio de 2009)

[12] http://rbtnxt.blogspot.com/ (9 de setiembre de 2009)

[13] http://nxtprograms.com/projects.html (7 de agosto de 2009)

ANNEXO A. Proyecto Ahorra luz

Con nuestro nuevo sensor y actuador ya montado, vamos a realizar un sencillo programa (**Fig A.1**) en que la NXT nos muestre por pantalla la resistencia de la LDR en cada momento, es decir, cada 3 ms, el tiempo que tarda en muestrear el conversor A/D.

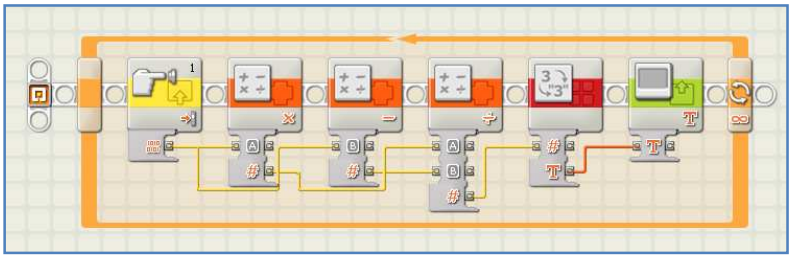

Fig A.1 Diagrama de bloques del programa LDR.

Lo siguiente que haremos será aprovechar nuestro pequeño programa para realizar unas medidas. ¿Cuál es el valor máximo de resistencia del LDR que obtenemos cuando está a oscuras? ¿Y el valor mínimo cuando está iluminado? Después de poner a oscuras nuestro LDR observamos que la máxima resistencia es de 60kΩ, mientras que, con la LDR iluminada, obtenemos una resistencia mínima de 1kΩ.

El siguiente objetivo es montar un robot que detecte la luz de una zona y que encienda el led gradualmente a medida que disminuya la cantidad de luz. Planteamos el siguiente programa:

Puesto que la R_{LDR} va de 1kΩ (iluminado) a 60kΩ (oscuridad), y que la potencia del LED la podemos regular de 0 (apagado) a 100% (máxima potencia), hemos creado una relación entre los rangos de R_{LDR} y la potencia del LED en saltos de 10 en 10. Esta correspondencia se muestra en la **Tabla A.1**.

Tabla A.1 Correspondencia entre R_{LDR} y la potencia del LED.

R_{LDR}	LED
60.000Ω	100%
54.100Ω	90%
48.200Ω	80%
42.300Ω	70%
36.400Ω	60%
30.500Ω	50%

24.600Ω	40%
18.700Ω	30%
12.800Ω	20%
6.900Ω	10%
1.000Ω	0%

Siguiendo esta tabla creamos un programa que calcule el valor de la resitencia del LDR y compare si está por encima de 6900Ω, a continuación si está por encima de 12.800Ω, etc., y vaya subiendo gradualmente la potencia del LED. Al probarlo nos encontramos con el primer problema:

- El LED nunca llega a encenderse ya que el valor del cálculo de la R_{LDR} nunca llega a 6900.

El problema reside en que Labview utiliza una escala de valores distinta a la de NXT-G y, por tanto, el valor Raw de su bloque del sensor de tacto devuelve valores mucho menores. Así que volvemos a realizar medidas aprovechando la funcionalidad de Labview “NXT Direct Commands” y esta vez observamos que el valor Raw varía entre 9 y 1014. Para este rango de valores, nos es mucho más fácil dividir esta medida entre 10 y realizar la tabla de valores de nuevo, solo que esta vez la relación será directa:

Tabla A.2 Nueva correspondencia entre el valor Raw y la potencia del LED.

Raw	**LED**
100	100%
90	90%
80	80%
70	70%
60	60%
50	50%
40	40%
30	30%
20	20%
10	10%
0	0%

En este punto del problema tenemos dos opciones para resolver-lo:

- Modificar el programa anterior siguiendo la misma estructura de comparación pero cambiando el rango de valores.
- Realizar un nuevo programa que utilice directamente el resultado de la división por 10 de Raw como entrada de potencia del LED (**Fig A.2**).

La segunda opción es sin duda la más cómoda, en cuanto a programación, además de que nos aporta una mayor sensibilidad al aparato, ya que el LED podrá funcionar con potencias del 43% o del 78%, es decir, que no sean múltiples de 10.

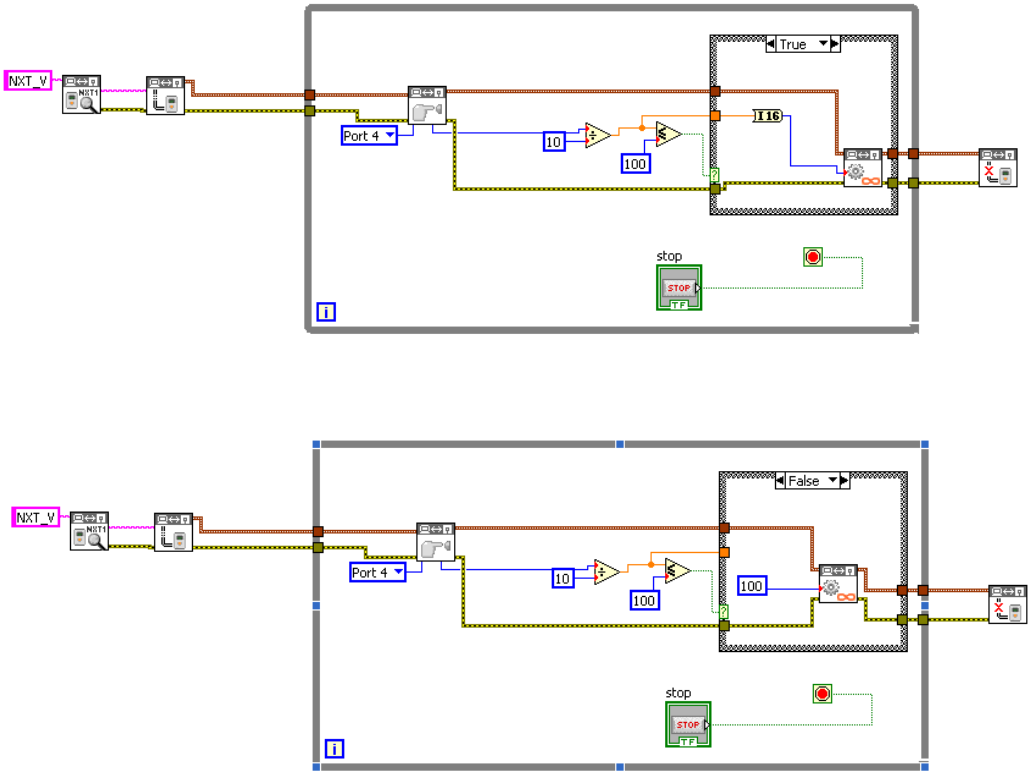

Fig A.2 Diagrama de bloques del programa Luz gradual.

Si llevamos a cabo la primera opción, en cambio, nos encontramos con un nuevo problema:

- A oscuridad máxima, el LED parpadea constantemente.

Esto se debe al planteamiento de nuestro programa. El programa compara de abajo a arriba si el valor está por encima de cada uno de nuestros rangos, es decir, compara primero si el valor es mayor a 10 y enciende el LED a esa potencia, a continuación comprueba si el valor es mayor a 20 y aumenta la potencia, cuando llega a potencia máxima se repite el bucle, así que la potencia vuelve a ser menor y va aumentando. Todo esto se traduce en un parpadeo constante que observamos y que puede producir dolor de cabeza, por tanto, nuestro diseño no sería apto para la aplicación. La forma de resolverlo es la siguiente:

- Modificar el programa para hacer la comparación de arriba hacia abajo.

Es decir, esta vez compararemos primero si es mayor que 100, en caso de serlo ajustaremos la potencia del LED a 100 y volverá a repetirse el bucle, y en caso contrario comprobaremos si es mayor que 90, y así sucesivamente.

Ahora que hemos logrado resolver nuestro objetivo, el siguiente paso será conseguir mantener siempre una determinada cantidad de luz medida con el LDR. Una posible aplicación sería la luz de un despacho; nosotros sabemos la cantidad mínima de luz que necesita recibir el usuario sobre su mesa para poder trabajar, pero es posible que la luz artificial que tenga sea excesiva teniendo en cuenta la luz natural que entra por la ventana. Este programa debería de ser capaz de regular la luz en función de su necesidad.

Para esta prueba, intentaremos mantener un nivel de luz mínimo que se traducirá en un valor de Raw máximo de 390. Cualquier valor de Raw mayor a ese requerirá de una aportación de luz por parte del LED para alcanzarlo. Hemos escogido este valor ya que es el valor mínimo de Raw que obtenemos con el LED a potencia máxima. De igual forma, si detectamos un valor de Raw inferior a 390, es decir, hay algún aporte de luz externa, disminuiremos la potencia del LED hasta llegar a este valor. Es decir, lo que vamos a construir será algo parecido a un comparador de histéresis (**Fig A.3**).

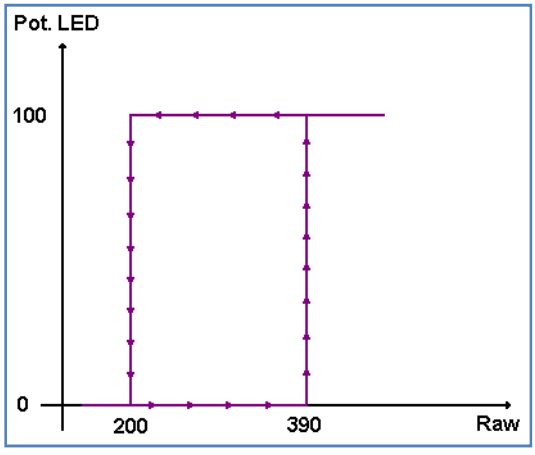

Fig A.3 Comparador de histéresis de nuestro programa Ahorra luz.

Puesto que la programación de este proyecto puede resultar complicada, vamos a plantear primero otro programa que nos servirá para simplificarla. Vamos a realizar una aplicación que nos permita encender la luz apretando el botón derecho de la NXT y disminuirla apretando el botón izquierdo, ambas acciones gradualmente (**Fig A.4**). Este planteamiento es el mismo que

utilizaremos más adelante para nuestro programa, con la diferencia de que la condición que hará que la luz aumente o disminuya no será un botón presionado, será nuestro sensor de luz.

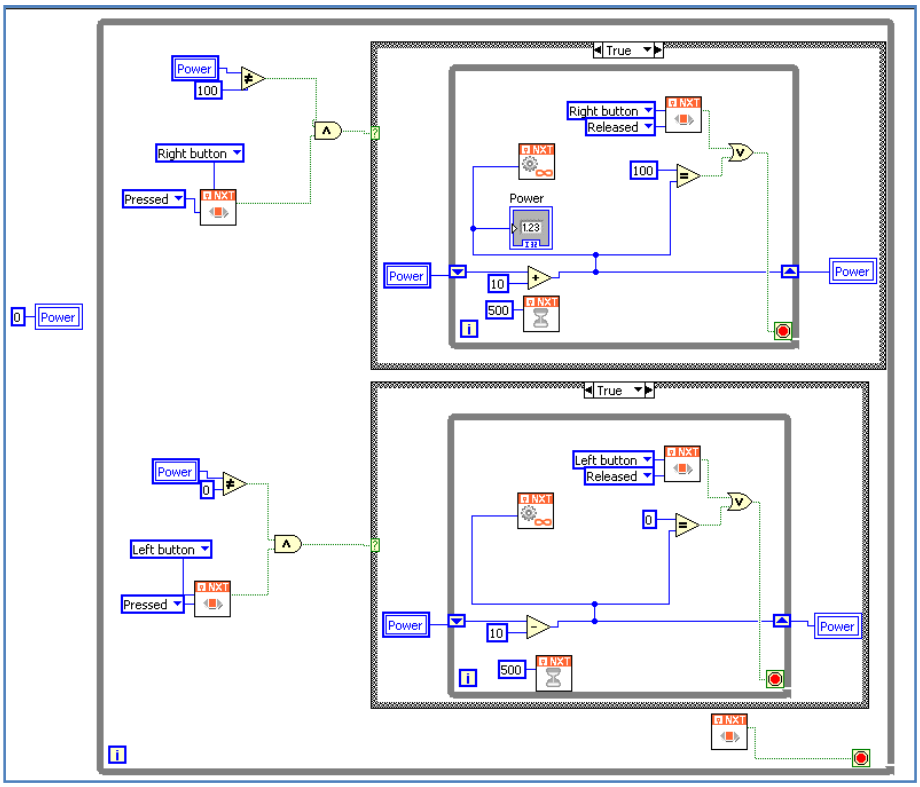

Fig A.4 Diagrama de bloques del programa Luz botones.

Inicializamos el valor de la potencia del LED a 0. Si presionamos el botón derecho de la NXT y la potencia es diferente a 100, aumentamos la potencia del LED de 10 en 10 cada 500ms hasta que, o bien llegamos a la potencia máxima (100), o bien soltamos el botón. Si presionamos el botón izquierdo de la NXT y la potencia es diferente de 0, disminuimos la potencia del LED de 10 en 10 cada 500ms hasta que, o bien llegamos a la potencia mínima (0), o bien soltamos el botón.

Siguiendo este planteamiento, modificamos el programa para conseguir nuestro objetivo inicial, utilizando nuestro sensor de luz (**Fig A.5**).

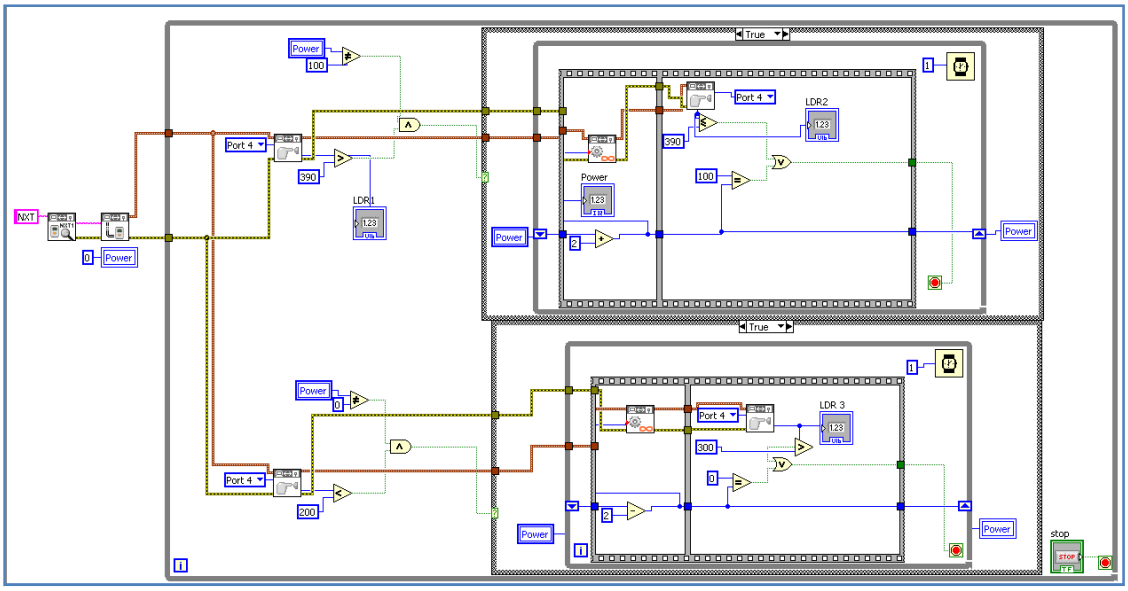

Fig A.5 Diagrama de bloques del programa Ahorra luz.

Inicializamos el valor de la potencia del LED a 0. Si el sensor de luz detecta un valor de Raw superior a 390 y la potencia es diferente a 100, aumentamos la potencia del LED de 2 en 2 cada 1ms hasta que, o bien llegamos a la potencia máxima (100), o bien detectamos un valor de Raw inferior o igual a 390. Si detectamos un valor de Raw inferior a 200 y la potencia es diferente de 0, disminuimos la potencia del LED de 2 en 2 cada 1ms hasta que, o bien llegamos a la potencia mínima (0), o bien detectamos un valor de Raw mayor a 300. Escogemos este valor de 300 en vez de 390, ya que un valor de Raw por encima de 390 entraría dentro de la condición anterior y se produciría un parpadeo molesto a la vista. También escogemos un valor de salto de 2% en vez del 10% ya que, un valor del 10% produce un cambio demasiado brusco en la detección de luz y, por tanto, siempre estamos muy por encima o muy por debajo del valor deseado y la lámpara parpadea.